LOMEditor 2005

Autores

Raúl Arriola Gómez

Susana de la Iglesia Arranz

Francisco Javier Piquer Sánchez

Índice de contenidos

SECCIÓN 1ª

INTRODUCCIÓN

1.1 Motivación

El surgimiento de Internet ha provocado la migración de la información y de los negocios a las redes informáticas. En particular en el ámbito de la Educación se han visto potenciadas las propuestas existentes para trasladar al marco de trabajo de un computador los elementos de la enseñanza presencial, conocidas genéricamente como *sistemas de enseñanza basados en computador*. En estas propuestas se pueden distinguir dos tendencias, unas que ponen énfasis en el hecho de conseguir una personalización del sistema muy sofisticada frente a la necesidad de mantenimiento del sistema e independencia de la tecnología usada, y otras que por el contrario, priman éste último aspecto sobre el primero. Dentro de esta última tendencia se encuentra una propuesta que plantea la creación de recursos educativos a partir de unidades mínimas de información reutilizables denominadas *objetos de aprendizaje*. Este planteamiento responde a la necesidad de reducir el tiempo de desarrollo y puesta en marcha de un sistema de enseñanza basado en computador. En este sentido uno de los objetivos es simplificar la creación de recursos a un simple ensamblaje de otros ya existentes que puedan estar localizados localmente (en el propio sistema) o remotamente (en otros sistemas interconectados mediante una red de telecomunicación), y construidos para plataformas heterogéneas.

Algunas de las características más importantes que presentan los sistemas basados en objetos de aprendizaje son:

1- Carácter modular de los objetos de aprendizaje.

Cada objeto de aprendizaje constituye una unidad de información independiente y completamente definida. Este hecho posibilita por una parte la *construcción incremental y modular* de los recursos educativos mediante agregación de objetos de aprendizaje básicos, y por otra la *encapsulación de la información* de los contenidos que representan.

2- Anotación semántica mediante metadatos estandarizados.

El contenido de un objeto de aprendizaje se describe mediante un conjunto de etiquetas normalizadas (metadatos) con capacidad expresiva, lo cual facilita:

- Aprendizaje adaptativo**.** El marcado de los objetos de aprendizaje con un nivel de granularidad suficientemente fino, permite una aproximación adaptativa del aprendizaje basada en asociar los metadatos del objeto con las características individuales del alumno o a los propósitos de las organizaciones que quieren usarlos. Además se puede obtener personalización mediante la *fijación posterior* (*late binding*) de un plan de estudios de acuerdo con las necesidades personales.

- Separación de sintaxis y semántica del contenido**.** El uso de metadatos establece una clara independencia entre el contenedor de la información (entidad software que sirve

para estructurar el contenido) y el contenido mismo (información introducida por el docente). Esta separación establece las bases para realizar con facilidad ciertas operaciones de procesamiento sobre los objetos de aprendizaje tales como modificación, búsqueda y gestión del contenido, sin tener que conocer sobre la materia representada.

3- Lenguajes de modelado educativo

Para organizar la información (contenidos y didáctica) de forma independiente a la que ésta será posteriormente utilizada (y de forma independiente de plataforma). Concretamente permiten desarrollar modelos, o plantillas, de diferentes actividades educativas, que pueden incluir exposiciones teóricas, guías de ejercicios y desarrollo de prácticas.

4- Naturaleza distribuida.

Los componentes de un sistema de enseñanza basado en objetos de aprendizaje puede que procedan de distintos proveedores conectados a través de una red de comunicaciones, requiriendo poder ser integrados de forma que se presente al usuario final una interfaz unificada. Otras características que presentan debidas a su naturaleza distribuida son:

- Operan en un entorno de una *red de comunicaciones*, en la que deben desplegar un repositorio de contenidos de aprendizaje extenso y escalable que facilite la integración y compartición de datos entre diferentes procesos, herramientas y aplicaciones.
- Deben ser *sistemas abiertos*, asegurando integraciones sencillas con sistemas de información existentes (legacy systems) tales como bases de datos o ERPs (Enterprise Resource Planning), así como otros sistemas LMS (Learning Management Systems) de terceros.
- El acceso a los recursos remotos debe ser tan sencillo como el acceso a los que se encuentran locales al sistema. De esta forma el nivel de abstracción se eleva al diseñar un sistema en base a objetos que pueden ya existir. En este sentido cada objeto de aprendizaje debería ofrecer una interfaz que defina sus características principales (*metainformación sobre el objeto***)**, una forma de darse a conocer en la red (*registro de objetos***)** y cómo poder localizarlos (*servicio de directorio distribuido*).

En paralelo con la aparición de los sistemas de enseñanza basados en objetos de aprendizaje, diferentes instituciones relacionadas con el ámbito de la educación han publicado *recomendaciones o especificaciones* sobre diversos aspectos relacionados con este tipo de sistemas. El objetivo perseguido ha sido definir un conjunto de estándares que normalicen el trabajo de toda la comunidad implicada en este ámbito, garantizando de esta forma la compatibilidad entre objetos desarrollados en un mismo contexto de enseñanza o en otros diferentes. La no existencia de tal compatibilidad hace difícil implementar *entornos de aprendizaje integrados*. Los estándares se pueden clasificar funcionalmente en tres tipos: *estándares acreditados* (desarrollados por grupos de trabajo dedicados en organizaciones internacionales como el LTSC de IEEE), *especificaciones* (descripciones

documentadas de soluciones que permiten su desarrollo e implementación como IMS) y *modelos de referencia* (describen guías concretas de implementación como ADL SCORM).El uso de estándares en los desarrollos ha permitido conseguir algunas ventajas tales como *flexibilidad* (configuración de los flujos de trabajo definidos de acuerdo a diferentes necesidades o paradigmas de aprendizaje), *interoperabilidad* (acordar especificaciones para el diseño, desarrollo y presentación de los contenidos pero permitiendo el intercambio y compartición de contenidos con otros sistemas de aprendizaje) o *extensibilidad* (integración simple de componentes adicionales al sistema).Por último destacar que el hecho de poder describir los objetos de aprendizaje a través de metadatos estandarizados junto al carácter modular que presentan y el establecimiento de un conjunto de estándares de uso común constituyen un conjunto de condiciones que facilitan la *reutilización* de los objetos en contextos y aplicaciones diferentes para los que fueron diseñados inicialmente.

1.2 Objetos de aprendizaje reutilizables

A nivel conceptual un objeto de aprendizaje consta de tres elementos: unos contenidos, unas descripciones del comportamiento del objeto, y un conjunto de metadatos que hacen referencia a los objetos. Aunque las implementaciones de un objeto de aprendizaje pueden ser muy variadas y pueden tener particularidades, todas tienen en común en implementar los objetos como unidades compuestas por un documento que describe los contenidos y cómo se relacionan, y los contenidos propiamente descritos por el documento. Así por ejemplo en el modelo de ADL SCORM:

1. Se han creado cuatro niveles de contenidos: *asset* (contenidos básicos de información tales como imágenes, páginas HTML, texto,…), *SCO y SCA* (agrupaciones de assets) y *Content Agregation* (agrupaciones de assets, SCOs y SCAs), y para cada uno de los tipos de contenidos se han definido metadatos estandarizados para describirlos: *Content Aggregation Meta-data, Activity Meta-data, SCO Meta-data, SCA Meta-data y Asset Meta-data***.**

2. Respecto a la implementación, se usan paquetes de contenidos basados en la especificación "IMS Content Packaging Information Model" formados por:

   - Un archivo XML denominado "imsmanifest.xml" que describe los contenidos y la organización de los mismos en el paquete, y que está dividido en 4 secciones: *Metadatos* (se describe el paquete en conjunto o alguno de sus componentes), *Organizaciones* (proporciona estructura a los contenidos),*Recursos* (permite describir recursos externos al paquete, los ficheros en el mismo o las relaciones entre los ficheros), y *Submanifest* (son archivos imsmanifest anidados que describen otras entidades dentro del paquete).
   - Los ficheros físicos descritos por el archivo XML.

Fig. 1.2.1 Objeto de aprendizaje según IMS.

El objetivo perseguido mediante la propuesta de los objetos de aprendizaje ha sido dar respuesta a cuatro cuestiones qué otras alternativas anteriores no han resuelto bien o ni siquiera han afrontado: *accesibilidad* (posibilidad de acceder y usar recursos educativos

desde cualquier lugar), *durabilidad* (posibilidad de soportar cambios tecnológicos sin tener rediseñar, reconfigurar o recodificar todo desde el principio), *interoperabilidad* (posibilidad de usar recursos desarrollados en cualquier lugar y realizados usando cualquier tipo de herramientas y para cualquier tipo de plataformas, con herramientas y plataformas distintas a las que sirvieron para realizar dichos recursos) y *reutilización* (flexibilidad para incorporar recursos en múltiples aplicaciones y contextos).

1.3 Generación de contenidos usando objetos de aprendizaje

Desde la perspectiva de la aplicación de las nuevas tecnologías a la enseñanza universitaria, en el futuro se pueden distinguir dos líneas de trabajo, por una parte está la interconexión de diferentes universidades que permita compartir materiales educativos propios de cada universidad y coordinar a las universidades para desarrollar conjuntamente otros nuevos que puedan ser ofrecidos a los alumnos para su realización desde cualquiera de ellas.

Y por otra está la producción de contenidos. En primer lugar los contenidos deberán mantenerse en formatos digitales fáciles de procesar y distribuir, que permitan ser reestructurados como requiera el profesor y puedan ser usados para estudiantes con distintas necesidades. Y en segundo lugar la generación de contenidos no deberá ser una labor exclusiva del profesor, sino que éste podrá apoyarse en materiales ya realizados por otros docentes e incluso por la experiencia y el trabajo del propio estudiante, pudiendo evolucionar los recursos de acuerdo a esta experiencia acumulada por el uso de los estudiantes.

Así pues en el contexto de los sistemas de enseñanza basados en objetos de aprendizaje, una de las actividades que se deben realizar es la *generación de los contenidos* de los recursos educativos de que constará el sistema de enseñanza. Los elementos básicos de partida de esta actividad son:

- Contenidos, que pueden estar anotados usando metadatos estandarizados.
- Objetos de aprendizaje realizados previamente para cierto entorno de aprendizaje.
- Estándares relacionados con esta actividad: *Metadatos* (para etiquetar de una forma consistente los recursos educativos digitales), *pedagógicos* (para disponer de un modelo de información y de asociación rico semánticamente, el cual describa el contenido y el procesamiento dentro de las unidades de aprendizaje desde una perspectiva pedagógica), *empaquetado de contenido* (para permitir a los recursos educativos ser transportados de un sistema a otro) y *comunicación de contenido* (para permitir incorporar reglas que describan el flujo de la instrucción a través del contenido según los resultados de las interacciones del usuario).
- Un conjunto de requisitos generados de forma independiente por diferentes actores que describen qué condiciones deberían satisfacer los objetos de aprendizaje desde la perspectiva particular del área de trabajo que representan cada uno de los actores. Estos requisitos influyen en la forma en que se deben relacionar y presentar los contenidos.

Los requisitos que pueden definirse son variables, como por ejemplo:

1. *Perfiles*. Es una descripción del tipo de alumno, y las características particulares del mismo.
2. *Objetivos.* Es una descripción formalizada de los conocimientos o niveles de aprendizaje que el alumno debe conseguir tanto en cada etapa del aprendizaje, como al final del mismo.

3. *Temporización /Planificación* .Tiene dos vertientes. Por una parte se trata de una descripción de los tiempos requeridos o estimados para llevar a cabo las diferentes etapas del aprendizaje, así como la totalidad del mismo. Y por otra parte representa las restricciones temporales para disponer de cada uno de los recursos educativos necesarios para poder generar el contenido.
4. *Estrategias de aprendizaje.* Es una descripción de la forma en que se quiere transmitir el conocimiento, y que los alumnos lleven a cabo el aprendizaje.
5. *Evaluación.* Es una descripción de la forma en que se quiere evaluar el aprendizaje llevado a cabo por el alumno.

El resultado de esta actividad debe ser un objeto o conjunto de objetos de aprendizaje ensamblados que satisfagan los requisitos establecidos. Para realizar esta actividad se deben llevar a cabo una serie de tareas:

- Analizar el grado de compatibilidad de los diferentes requisitos planteados, y estudiar la viabilidad de poder generar objetos de aprendizaje que los cumplan en una proporción alta. Así para tomar la decisión de llevar a cabo o no el desarrollo, será necesario estimar el coste de realización de los objetos de aprendizaje que serían necesarios.
- Si la estimación resulta positiva, analizar si existen contenidos u objetos de aprendizaje locales, que permitan satisfacer los requisitos planteados. En caso de no existir buscar en sistemas remotos conectados a través de una red de comunicaciones. Si esta búsqueda diera resultados positivos, se debería comparar el coste de adquisición de los objetos remotos frente al coste estimado de llevar a cabo su desarrollo explícito, eligiendo la solución más beneficiosa.
- Si las búsquedas resultarán infructuosas, se debería estudiar la posibilidad de generar dicho contenido de forma explícita estimando el coste de dicho desarrollo, y por tanto su viabilidad.
- En caso de disponer de contenidos u objetos de aprendizaje, analizar cómo se llevará a cabo la composición de los objetos de aprendizaje.
- Realizados los objetos de aprendizaje, llevar a cabo un estudio de la calidad de los objetos generados.

1.4 Problemática en la Generación de contenidos

El problema general que se plantea en la generación de contenidos, es el problema de cómo generar contenidos a partir de un conjunto de objetos de aprendizaje producidos anteriormente, y que además los contenidos que se generen cumplan un conjunto de requisitos especificados. Este problema se puede desglosar en varios subproblemas:

1. *Problema de representación de la información.*

Los requisitos que deben verificar los contenidos se especifican de forma independiente por cada uno de los actores y deben estar descritas en algún lenguaje, sobre el cual sea posible llevar a cabo los razonamientos posteriores. Es razonable que estos lenguajes en general no sean los mismos, ya que en cada ámbito se necesitarán unos elementos expresivos y unos simbolismos particulares para poder representar lo mejor posible los requisitos referidos a dicho ámbito en particular.

En este sentido se plantea la tarea de analizar los lenguajes ya definidos para representar requisitos en los diferentes ámbitos, y comprobar si son adecuados. En caso de no serlo se debe estudiar la adaptación de los mismos o la definición de nuevos lenguajes.

2. *Problema de procesamiento de la información.*

Suponiendo representados los requisitos en sus respectivos lenguajes, será necesario procesar las representaciones para poder razonar si los requisitos son satisfactibles en conjunto, o si no lo son en que grado y cuales se pueden satisfacer. En cualquiera de los casos en que se pueda satisfacer algún requisito se deberá tener como resultado qué contenidos u objetos de aprendizaje son necesarios, cómo buscarlos y cómo ensamblarlos.

En este sentido se pueden definir las siguientes tareas a realizar:

a. Analizar la definición de un lenguaje común al cual traducir las representaciones de los requisitos de cada ámbito, descritos en principio usando diferentes lenguajes.
b. Analizar qué métodos de razonamiento se pueden usar para procesar los requisitos planteados.
c. Analizar la definición de un lenguaje para representar los resultados del procesamiento.

3. *Problema de recuperación y composición de la información.*

A partir de los resultados del procesamiento de los requisitos, se deben recuperar los contenidos u objetos de aprendizaje requeridos, y llevar a cabo la composición de los mismos.

En este sentido se pueden definir las siguientes tareas a realizar:

a. Analizar cómo llevar a cabo la búsqueda y recuperación de contenidos u objetos de aprendizaje locales o remotos.
b. Analizar la forma de componer o adaptar contenidos u objetos de aprendizaje diseñados específicamente para el mismo entorno de aprendizaje o para otros diferentes.

4. Problema de estimación.

Llevar a cabo una generación de contenidos plantea la necesidad de estimar dos aspectos:

1. Estimación del coste.
   Se trata de una estimación a priori. El desarrollo de los objetos de aprendizaje que es necesario realizar, resultado de procesar los requisitos definidos, va a conllevar un coste en esfuerzo de tiempo y recursos empleados. Es por ello que es necesario disponer de métricas con las que poder estimar dicho coste, de forma que se pueda plantear la viabilidad o no del desarrollo de dichos objetos.

2. Estimación de la calidad.
   Se trata de una estimación a posteriori. Los objetos de aprendizaje que se desarrollen, deberán ser sometidos a unos controles de calidad, que establezcan el nivel alcanzado en los mismos. Para ello se ha de disponer de una adaptación concreta de los estándares de calidad generales al ámbito de los objetos de aprendizaje, con la cual poder definir un conjunto de atributos, susceptibles de ser medidos, y unas métricas para poder medirlos.

En este sentido la principal tarea a realizar consiste en analizar si existen modelos de estimación del coste de desarrollo de objetos de aprendizaje, y especificaciones concretas sobre la calidad en los objetos de aprendizaje. En caso de no existir se deberá estudiar su definición. Observar que además de las estimaciones mencionadas, otra actividad que es necesario realizar es una planificación que defina temporalmente mediante hitos y diagramas, las fases en que se debe llevar a cabo el desarrollo de los contenidos, qué tiempo dura cada fase o que actividades están relacionadas y son necesarias terminar para llevar a cabo otras. La existencia de una planificación permite organizar el desarrollo de los objetos de aprendizaje, distribuir los recursos e introducir rectificaciones en la misma ante retrasos que se puedan producir en la terminación de las diferentes fases en que se divide su desarrollo.

1.5 Escenario real de Generación de Contenidos

Se va a describir un escenario real de la actividad de la generación de contenidos.

En esta actividad intervienen tres actores principales:

- *Ingeniero del Software*→ Es una persona experta en el aspecto técnico e informático, y que no tiene porqué conocer nada sobre la materia en que versará el aprendizaje. Así por ejemplo instancias de este actor, serían:
  Planificador*.*
  Estima el tiempo necesario para llevar a cabo el objeto de aprendizaje que han especificado los especialistas. En concreto debe establecer el tiempo que consumirá en generar objetos que no estén hechos previamente.
  **Estimador del esfuerzo.*
  Estima el esfuerzo necesario en personal, y coste económico que habrá que realizar para generar los contenidos. También debería establecer la viabilidad o no de dicho esfuerzo.
  **Aseguramiento de la calidad.*
  Estima la calidad de los objetos de aprendizaje que se realicen, en base a ciertos parámetros tales como reusabilidad, granularidad,…

- *Especialista*→ Es una persona experta en cierta materia, la cual va a especificar los requisitos que debería tener el objeto de aprendizaje desde el punto de vista de la materia de la cual es experta. Así por ejemplo instancias de este actor, serían :
  **Experto en los contenidos*. (¿Qué se quiere enseñar?)
  Fuente de información de los contenidos de los que se quiere que trate el aprendizaje.
  Evaluador*.* (¿Cómo se quiere evaluar el aprendizaje?)
  Establece las pautas para llevar a cabo la evaluación de los contenidos,
  **Experto en psicopedagogía*. (¿Qué metodología de enseñanza se quiere usar para realizar el aprendizaje?)
  Indica que metodología es la mejor para llevar a cabo el aprendizaje (castigo-premio, repetición, ensayo-error…).
  Secuenciador*.* **(¿**Qué hitos y en cuánto tiempo se quiere que se lleve el aprendizaje?)
  Establece los hitos u objetivos y el tiempo en que deben alcanzarse, así como el tiempo global para finalizar el aprendizaje.
  **Perfilador*. (¿Qué perfiles y características tienen los alumnos que van a llevar a cabo el aprendizaje?)
  Describe las características de los alumnos a los que va dirigido el aprendizaje.

- *Productor*→Es la persona que financia la generación de los contenidos, y la cual va a establecer los requisitos en cuanto a tiempo de ejecución, personal dispuesto a contratar para su realización así como el coste económico que está dispuesto asumir.

- *Sistema generador de contenidos*→ Es el sistema de autoría que debe partir de los requisitos que han especificado cada uno de los especialistas, y que bajo la supervisión de los ingenieros del software, deberá establecer qué coste y esfuerzo supone, y si es viable

crear un objeto de aprendizaje con las características que se piden. En caso afirmativo deberá llevar a cabo la búsqueda, creación y composición de los objetos de aprendizaje y contenidos que sean necesarios para crear el objeto pedido. Además una vez creado el objeto llevará a cabo un análisis de la calidad del mismo.

En caso negativo deberá informar de las causas por las que no es viable su realización.

A la vista de los actores descritos se plantea el siguiente escenario:

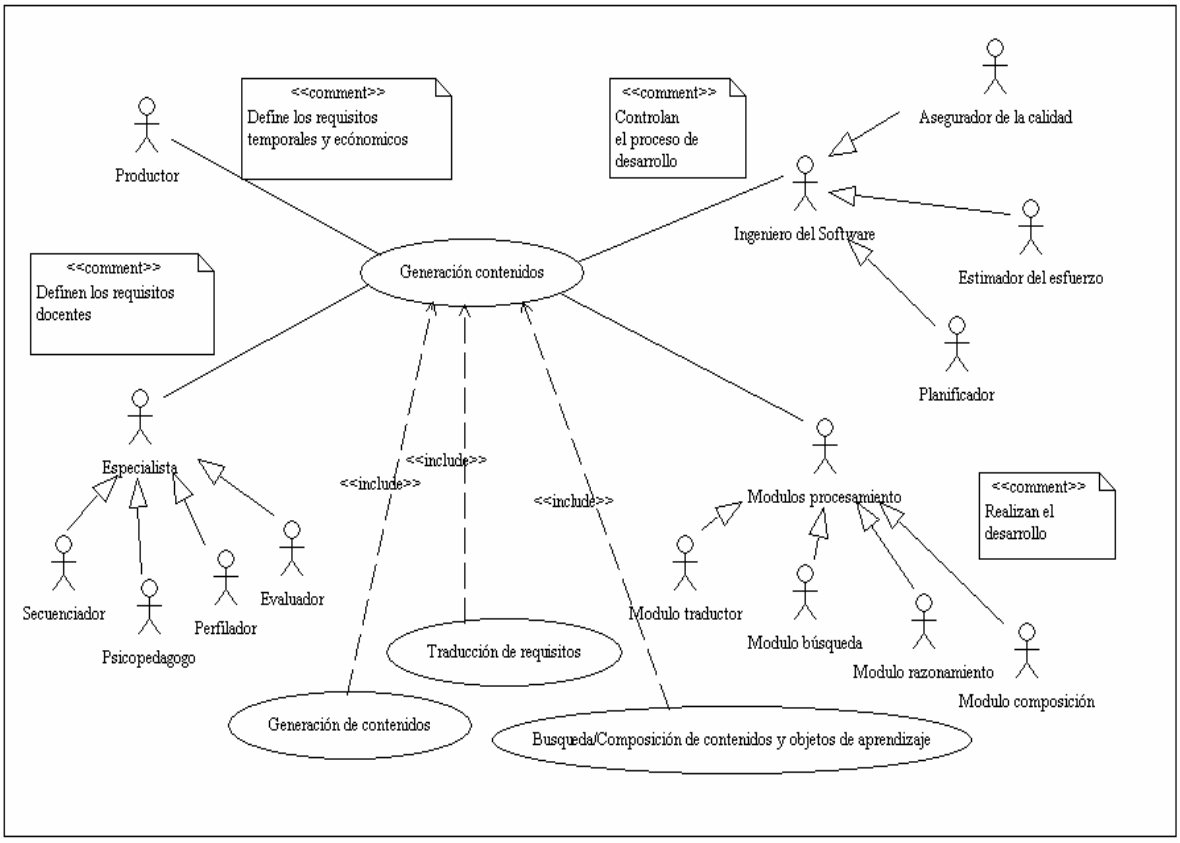


Fig. 1.51 Escenario de trabajo

Fijada una materia de aprendizaje, las condiciones del productor y unos objetivos sobre la forma de evaluación, temporización del aprendizaje, contenidos a aprender, perfiles de los alumnos y metodología para llevar a cabo el aprendizaje, se puede enumerar el flujo de actividades que se deben llevar a cabo:

a) Cada uno de los especialistas deberá expresar en un documento y, usando los estándares propios de su campo de trabajo, cuales son los requisitos que deberían cumplir los contenidos que se generen. En el caso particular del experto en los contenidos, además de establecer qué contenidos se deberían aprender, también deberá introducir algunos de los contenidos en el sistema y usar alguna técnica de anotación para etiquetarlos de acuerdo a los metadatos que se hayan establecido. Resultado de esta actividad es disponer de un conjunto de documentos (cada uno de

ellos escrito en un lenguaje diferente) de los requisitos que debe cumplir el objeto de aprendizaje. Otra entrada del sistema, serán las condiciones que establezca el productor.

b) Los ingenieros del software deberán estimar el coste temporal, en esfuerzo y económico que supone llevar a cabo un objeto de aprendizaje de las características reflejadas en los documentos. Además el sistema debe establecer la compatibilidad de los requisitos expresados. Es posible que los requisitos que cada uno de los especialistas solicita, sean incompatibles de llevarlos a cabo todo a la vez, y haya que establecer unas prioridades. En función de estos análisis y de las condiciones establecidas por el productor el sistema debe juzgar si es viable o no la realización del objeto de aprendizaje. En caso negativo deberá pararse el proceso de generación de contenidos expresando las causas de la inviabilidad, y en caso positivo pasar a realizar la siguiente actividad.

c) Habiendo resultado positivo el estudio de viabilidad, a partir de los documentos de requisitos, el sistema debe establecer qué contenidos u objetos de aprendizaje son necesarios para generar el objeto solicitado. Conocidos los contenidos y objetos necesarios, llevará a cabo una búsqueda local y en remoto de los mismos. Se pueden plantear varias situaciones. Un caso es que algunos de los objetos y contenidos buscados se encuentren en repositorios remotos. En este caso se deberá hacer un análisis para comparar el coste de obtención de dicho objeto remoto frente al coste de llevarlo a cabo. También podría plantearse el caso de que ambos costes estuvieran por encima de las condiciones establecidas por el productor, debiendo dar por inviable la generación de los contenidos. Otro caso es la situación de no encontrar alguno de ellos ni remota ni localmente. Aquí de nuevo se planteará un estudio de la viabilidad de desarrollar los objetos no encontrados, estimando para ello el coste temporal, económico y esfuerzo. Si en alguno de los casos establecidos fuera inviable la generación de los contenidos necesarios, se parará el proceso de generación de contenidos y se expresará las causas de la inviabilidad. En cualquier otro caso se pasaría a realizar la siguiente actividad.

d) Obtenidos los objetos y contenidos necesarios para generar el contenido, se pasará a componerlos y obtener el objeto de aprendizaje requerido.

e) En último lugar se llevara a cabo diferentes medidas para estimar la calidad del objeto obtenido.

A la vista de las actividades descritas se plantea el siguiente flujo de trabajo:

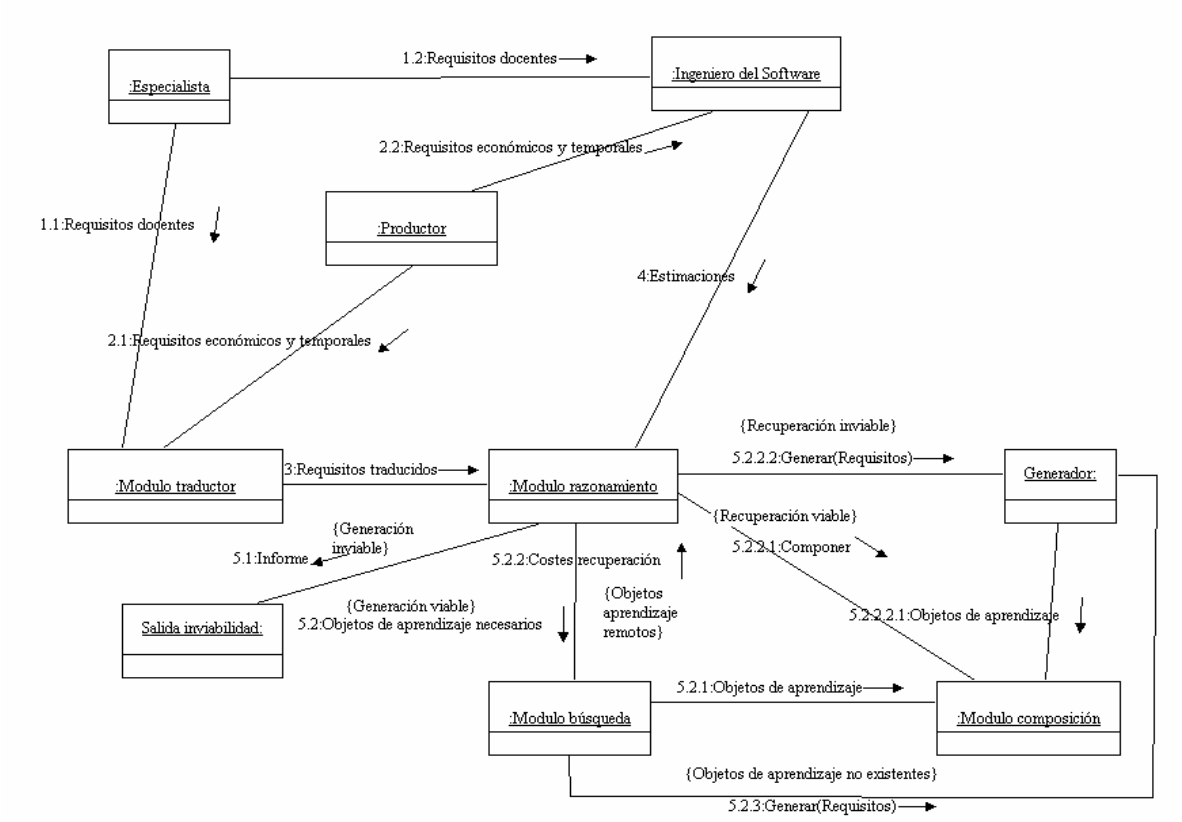


Fig. 1.5.2 Escenario de trabajo

SECCIÓN 2ª

EL PROYECTO LOMEDITOR

2.1 Planteamiento del proyecto

El proyecto LOMEditor surge de la necesidad de creación de objetos de aprendizaje, dentro del campo de la enseñanza basada en las nuevas tecnologías. Su principal objetivo es constituir un entorno de creación, edición y utilización de objetos de aprendizaje que complete y mejore el uso de estos últimos, facilitando de esta forma su aplicación en entornos académicos. Para ello era primordial independizar la herramienta de las diferentes gramáticas que aplican los estándares.

Con este fin se ha creado la herramienta LOMEditor 2005, capaz de abrir cualquier tipo de objeto de aprendizaje generado correctamente a partir de una gramática, permitiendo ampliar su campo de trabajo. Por supuesto ofrece la posibilidad de comenzar un objeto de aprendizaje desde cero, con lo que evita la necesidad de contar con alguna referencia.

Pero el proyecto LOMEditor va mucho más allá en sus objetivos. La herramienta de autoría proporciona toda la funcionalidad necesaria para que cualquier desarrollador de e-learning pueda trabajar con ella, pero esto no es todo lo que puede ofrecer a sus usuarios. LOMEditor cuenta con diversos módulos complementarios que aumentan la funcionalidad del mismo, colaborando intensamente en la mejoría de la creación de objetos de aprendizaje. Estos módulos, que más adelante serán expuestos de forma más precisa y completa, son el módulo de evaluación de la calidad de objetos de aprendizaje y el módulo de composición de objetos.

Inicialmente, LOMEditor se había pensado como un proyecto que acercase un poco más el e-learning al mundo práctico de la enseñanza, pero gracias a estas funcionalidades avanzadas que exploran campos hasta entonces poco experimentados, se ha situado como referente para futuros desarrollos y avances del sector. Actualmente LOMEditor es una herramienta multifuncional con facilidades prácticamente inéditas en su campo, que cumple con creces los objetivos planteados al inicio del proyecto.

La herramienta de autoría de objetos de aprendizaje LOMEditor es una aplicación local para el trabajo independiente y personal del usuario, en cuanto a la creación o modificación de objetos de aprendizaje. Cuenta con una amigable interfaz de usuario en Java, de muy sencilla utilización, que contribuye a facilitar el trabajo del usuario de la herramienta. Cuenta también con una base de datos en MySQL donde guardamos información relativa a la evaluación de calidad, desde un punto de vista de razonamiento basado en casos. Se almacenan, por tanto, los casos útiles para el usuario en un futuro.

La interacción se realiza, por tanto, a nivel local. Sin embargo, entre sus aplicaciones existe la posibilidad de utilizar LOMEditor como soporte en exposición de contenidos, gracias a su capacidad audiovisual.

2.2 Puntos clave de investigación

Es cierto que no existen demasiados sistemas o entornos que permitan la creación y el trabajo con objetos de aprendizaje; no obstante hay algunos proyectos de software libre que ofrecen un nivel francamente aceptable. Por otro lado, algunas casas de importante prestigio, como Microsoft, ofrecen herramientas de desarrollo en este sector.

Algunos de ellos, como hemos dicho, si cuentan con herramientas útiles y completas, que permiten un trabajo satisfactorio en este campo. Sin embargo LOMEditor se desmarca ofreciendo algunas novedades e innovaciones que hacen de esta herramienta un referente importante en el sector, ya que abre puertas a futuros proyectos que verdaderamente supongan el impulso definitivo del e-learning.

Entre los puntos clave a los que nos referimos destacan principalmente los siguientes:

1. *Independencia de la gramática*

   Desde el principio, este ha sido el principal objetivo con el que ha sido planteado el proyecto LOMEditor. La capacidad de trabajar cualquier tipo de objeto de aprendizaje, independientemente de la gramática que utilicen (siempre dentro de unos estándares), supone posiblemente el mayor avance que la herramienta aporta al sector.

   Para lograr este fin, la aplicación debe evaluar la corrección del manifiesto que acompaña a la gramática, y que determina el objeto de aprendizaje. Para ello se cuenta con un paquete de clases Java implementadas con este fin.

   Una vez validado el objeto, permite editar el mismo a partir de la gramática procesada, pues es la única forma de garantizar que mantendrá el estándar original. Actualmente es prácticamente la única herramienta que ofrece esta posibilidad.

   Esta innovación realmente constituye una ventaja enorme para una herramienta de estas características, pues no solo permite el trabajo con un más amplio conjunto de objetos de aprendizaje, sino que protege a LOMEditor de quedar aislada u obsoleta ante la aparición de gramáticas más modernas dentro de los estándares.

2. *Evaluación de la calidad*

   Este primer punto es una facilidad que LOMEditor ofrece a sus usuarios, y cuyo objetivo no es otro que colaborar con estos a un mejor desarrollo del objeto de aprendizaje que estén tratando.

LOMEditor, gracias a un conjunto de casos y un razonamiento basado en conocimiento adquirido, permite al usuario evaluar la calidad de un objeto, pudiendo de esta forma comprobar si se encuentra o no en un buen camino. Es además una forma muy cómoda de etiquetar y comparar objetos, pues existe la posibilidad de asignar la calidad manualmente, es decir, categorizar dichos objetos.

Se trata por tanto de una importante innovación en el sector, que hasta entonces no había sido llevada a cabo de forma satisfactoria, y a la vez es una novedad que invita a la colaboración entre aplicaciones e-learning y sistemas basados en conocimiento.

3. Composición de objetos de aprendizaje

Destacamos también esta última innovación dentro del campo de creación de objetos de aprendizaje. Se trata de la posibilidad de componer un objeto a partir de objetos más sencillos, o simplemente complementarios.

Las posibilidades abiertas por esta facilidad son muy importantes, ya que supone la capacidad de fusión de conocimientos para lograr un mayor campo de actuación, es decir, partiendo de pequeños objetos de aprendizaje básicos, crear un completo curso que abarque todos estos pequeños módulos.

Un ejemplo de aplicación sería el desarrollo de pequeños paquetes de objetos de aprendizaje que contienen información mínima reutilizable. Contando con estos paquetes, cualquier tutor o desarrollador puede formar sus cursos mediante composición de esta información. Así, un profesor de matemáticas, por ejemplo, estructuraría su curso en temas, donde cada tema es un objeto de aprendizaje compuesto por pequeños objetos que contienen información concreta y reducida de determinado ámbito matemático. El curso sería el objeto de aprendizaje que compusiera estos temas. Tenemos por tanto un objeto de aprendizaje fácilmente realizado gracias a la composición.

Colaborar simplificando la creación de dichos objetos, como hace la composición, ha sido un objetivo sobre el que se han realizado importantes esfuerzos. Puede decirse que LOMEditor permite al usuario básico abstraerse, de esta forma, del más o menos complejo nivel de creación de los objetos básicos.

Por todo esto creemos que LOMEditor es una completa herramienta que no solo cumple con los objetivos principales de las aplicaciones del sector, sino que ofrece un conjunto de mejoras que marcan una diferencia notable respecto a las anteriores.

SECCIÓN 3ª

ESPECIFICACIÓN DE REQUISITOS DEL SISTEMA

3.1 Introducción

3.1.1 Descripción general

Como ya se ha comentado previamente el objetivo de este proyecto era el de desarrollar una herramienta de autoría de objetos de aprendizaje, respetando en todo momento la línea que marcan los estándares de e-learning al respecto.

La motivación principal para llevar a cabo la realización de este proyecto ha sido la de intentar crear una herramienta de autoría que marque la diferencia con respecto a las existentes en el mercado, sobre todo en una cuestión muy importante: los estándares.

La mayoría de las herramientas de autoría que existen en el mercado en la actualidad basan su funcionamiento en una gramática concreta, especificada en el estándar correspondiente, y que se puede modificar, en algunos casos, por otra que ofrezca el sistema. No obstante, y debido al continuo progreso en el campo del e-learning, estas gramáticas se encuentran en continuo cambio, y la aparición de nuevos estándares cada cierto tiempo está garantizada. Debido a este continuo avance las herramientas de autoría quedan obsoletas rápidamente al no permitir la incorporación de los nuevos estándares que ofrece el mercado del e-learning.

Por lo tanto, una herramienta de autoría que fuese independiente de la gramática que utilice el objeto de aprendizaje supone una mejora sustancial con respecto al mercado actual.

LOMEditor 2005 se ha desarrollado de esta forma. Realiza la evaluación del manifiesto asociado al objeto de aprendizaje analizando si el manifiesto cumple la estructura definida en su gramática. Cada objeto de aprendizaje llevará asociados tanto su manifiesto como su gramática, por lo que si aparece una nueva gramática la validación se realizará con respecto a ésta y el sistema se auto-actualizará conforme a los avances en los estándares de e-learning.

3.1.2 Ámbito de la aplicación

El sistema pretende ofrecer las funcionalidades básicas de las herramientas de autoría, sin limitar su comportamiento al uso de una gramática específica, sino ofreciendo al usuario un sistema que se adapta a los cambios sufridos por estos estándares.

Además de las funcionalidades básicas que soportan la mayoría de las herramientas en el mercado, el sistema contendrá dos módulos que permitirán realizar un tratamiento avanzado de los objetos de aprendizaje.

Estos dos módulos adicionales del sistema serán: un módulo que permitirá la composición de objetos de aprendizaje, fusionándolos para crear objetos más complejos a partir de los originales; y un módulo de evaluación de calidad de objetos, que permitirá, utilizando estrategias de inteligencia artificial, evaluar de forma inteligente un objeto de aprendizaje basándose en el contenido de sus metadatos.

3.1.3 Producto a entregar

El producto a entregar consta de 3 módulos principales para el tratamiento de objetos de aprendizaje:

Módulo 1: Herramienta de autoría de objetos de aprendizaje.
Módulo 2: Módulo de composición de objetos de aprendizaje.
Módulo 3: Módulo de evaluación de calidad de objetos de aprendizaje.

Por tanto, las funcionalidades del sistema serán divididas según el módulo al que pertenecen.

Funcionalidades de la herramienta de autoría

1. Apertura de objeto de aprendizaje previa validación.
    1.1. Visualización del contenido del objeto de aprendizaje.

2. Edición de un objeto de aprendizaje.
2.1. Adición de nuevos elementos.
2.2. Eliminación de elementos.
2.3. Modificación de atributos de elementos.

3. Guardar el manifiesto (imsmanifest.xml) del objeto de aprendizaje con el que se está trabajando.

4. Guardar un objeto de aprendizaje abierto en forma comprimida.

5. Creación de un nuevo objeto de aprendizaje.

6. Ayuda

7. Consultas sobre una base de datos que contiene información de objetos de aprendizaje.

Funcionalidades del módulo de composición

1. Mostrar objetos de aprendizaje previa validación.

2. Composición en paralelo de dos objetos de aprendizaje.

3. Composición en profundidad de dos objetos de aprendizaje.

4. Guardar objetos compuestos en forma comprimida.

5. Guardar todos los objetos compuestos hasta el momento.

6. Salir de la composición.

Funcionalidades del módulo de evaluación de calidad

1. Evaluación manual de la calidad de objetos de aprendizaje abiertos y almacenamiento persistente en una base de datos.

2. Clasificación automática de objetos de aprendizaje de acuerdo a los mismos atributos usados en su evaluación.

3.2 Contexto de uso

3.2.1 Perfiles de usuario

Dentro del conjunto de usuarios potenciales de esta herramienta debemos distinguir principalmente entre tres de ellos: el tutor, el desarrollador e-learning y el alumno especializado.

1. *Usuario Tutor*

   Este usuario recurrirá a LOMEditor en busca de un apoyo para impartir sus conocimientos.

   LOMEditor ofrece la posibilidad de abrir y visualizar cualquier tipo de objeto de aprendizaje correctamente construido, lo que lo convierte en una herramienta de elevada utilidad a la hora de realizar un soporte en enseñanza a cualquier tutor.

   Dadas las características que presenta un objeto de aprendizaje, parece verdaderamente adecuado para su utilización en campos de enseñanza donde el componente audiovisual tiene un peso relevante. La posibilidad de mostrar, vídeos, documentos, imágenes, páginas relacionadas, etc., facilita en gran medida la labor de este tutor, haciendo llegar sus conocimientos de forma más sencilla y amigable a sus alumnos.

2. *Desarrollador e-learning:*

   Se trata de un usuario básico para la propagación de los objetos de aprendizaje, ya que será el encargado de crear dichos objetos.

   LOMEditor le ofrece la posibilidad de crear a partir de cero, o a partir objetos de aprendizaje ya desarrollados, nuevos objetos de aprendizaje.

   Es aquí donde sale a relucir la principal ventaja de LOMEditor sobre sus competidores, y es el soporte que ofrece a la creación de los objetos. El desarrollador dispone de la funcionalidad habitual y necesaria para generar el objeto, pero además cuenta con dos módulos exclusivos de esta herramienta, incluidos para este fin: la evaluación de la calidad del objeto y la composición de objetos nuevos a partir de varios objetos ya creados.

   La evaluación de la calidad ayuda inestimablemente al desarrollador, ya que es una forma de contrastar las virtudes y defectos de su creación. Permite

además plantearse objetivos de calidad, para garantizar objetos de aprendizaje con un buen nivel y completitud.

Por otro lado, la posibilidad novedosa de poder construir objetos de aprendizaje por composición de otros objetos, abre el campo del desarrollo de objetos de aprendizaje. Al igual que en otros campos de la ciencia, la capacidad de encapsular e independizar los desarrollos es un aspecto muy importante, ya que permite la reutilización de dichos elementos. En este caso la ventaja es evidente, pues podremos ofrecer paquetes de objetos de aprendizaje con información básica, para que cada desarrollador componga su objeto completo de la forma que mejor crea.

Es por tanto el usuario desarrollador e-learning el que más ventajas encontrará con el uso de LOMEditor.

3. *Alumno especializado*

El alumno especializado es un usuario que utiliza los servicios de LOMEditor en busca de una información concreta.

En muchos casos la figura del tutor no es necesaria, o conlleva unas connotaciones más académicas. En este caso se trata de ofrecer al usuario unos conocimientos perfectamente empaquetados y listos para su asimilación.

Es responsabilidad del desarrollador ofrecer estos objetos, pero será el usuario especializado el que más partido saque de ellos. La posibilidad de recurrir a LOMEditor para realizar cursos avanzados rápidos va más allá de las necesidades académicas.

Un alumno especializado no necesita completos manuales, ni cursos de varias semanas, pues busca información concreta y bien definida. Esto lo vemos día a día en las empresas, que tratan de acumular la mayor cantidad de información posible en extensos documentos para ofrecérsela a sus empleados, los cuales deben pasar horas tratando de localizar lo que realmente es útil. LOMEditor, sin embargo, permite organizar esta información de forma mucho más concreta, y además de forma interactiva.

En resumen, el usuario especializado busca en LOMEditor cursos rápidos y concretos, ahorrando con ello tiempo y mejorando su formación.

3.2.2 Usos del software

Como se ha dejado entrever en el punto anterior, las aplicaciones de LOMEditor son numerosas, principalmente centradas en el campo de la enseñanza, pero no necesariamente aisladas en él.

Entre los usos que pueden dársele a esta herramienta destacamos los siguientes, atendiendo a su practicidad y demanda:

1. *Soporte en enseñanza*

   Este uso será principalmente aplicado por profesores o tutores, que deseen completar su exposición con algún objeto de aprendizaje determinado. El carácter audiovisual de LOMEditor colabora a facilitar su utilización en estos fines.

   Existe la posibilidad incluso de hacer que LOMEditor lleve el peso del curso, pues su capacidad de manejo de todo tipo de recursos así lo permite.

2. *Cursos rápidos especializados*

   Estos cursos están enfocados a una utilización más personal y determinada. Vamos en busca de conocimientos muy específicos, y deseamos adquirirlos nosotros mismos mediante la herramienta.

   Ampliando esta idea podemos pensar en cursos adaptados a ciertos cargos, o ciertas personas, por ejemplo, desde un curso rápido de programación en Java, hasta una introducción al manejo de aeroplanos.

3. *Organizador de conocimientos*

   La necesidad de almacenar y organizar conocimientos es una de las bases de la informática. El problema suele presentarse a la hora de recuperar estos conocimientos y la forma en la que pueden usarse. LOMEditor permite organizar y recuperar toda la información guardada en los objetos de aprendizaje, así trabajar directamente sobre la misma.

4. *Presentación de contenidos*

   De nuevo es su enfoque audiovisual el que presenta aplicaciones muy útiles. En este caso se trata de realizar presentaciones en grupo. Una buena gestión de

los recursos de LOMEditor puede ayudar a este fin, ya que permite al ponente ayudarse de la herramienta para hacer llegar fácilmente los contenidos.

5. *Ocio*

Aunque es una herramienta principalmente enfocada a enseñanza y conocimientos, no hay que olvidar aspectos de gran demanda en nuestra sociedad, como son el ocio y el entretenimiento. Está en manos del desarrollador poder generar contenidos en este sentido, registrando información que trate aspectos menos formales.

En resumen podemos decir que LOMEditor ofrece un amplio campo de aplicación, resultando especialmente útil en aquellos sectores para los que ha sido diseñado.

3.3 Servicios – Requisitos funcionales

3.3.1 Servicios básicos

Módulo 1: Herramienta de autoría

1. Apertura de objetos de aprendizaje

El sistema será capaz de abrir objetos de aprendizaje. Previa a esta apertura la herramienta realizará una validación del manifiesto del objeto de aprendizaje (imsmanifest.xml) frente a su esquema de manifiesto (imscp_v1p1.xsd) y a su esquema de metadatos (imsmd_v1p2p2.xsd).

Los objetos deben haber sido previamente zipeados y contener en su directorio raíz los ficheros correspondientes a su manifiesto, esquema de manifiesto y esquema de metadatos, y serán escogidos mediante una ventana de navegación. Se solicitará también al usuario, mediante una ventana de navegación, la ubicación de los ficheros que serán descomprimidos para poder trabajar con el objeto de aprendizaje.

Después de la descompresión de los archivos del objeto de aprendizaje, se procederá a la validación del mismo.

Si el objeto supera esta validación, entonces se visualiza en un árbol de navegación que se muestra en el panel izquierdo de la herramienta (en caso contrario no se permite su visualización) y se presenta el contenido del mismo mediante un panel en el centro de la pantalla. El árbol representa en sus ramificaciones y nodos la jerarquía de metadatos que define al objeto de aprendizaje, la cual es extraída del documento imsmanifest.xml. De este mismo documento se extrae la información necesaria para mostrar el contenido de cada una de las organizaciones del objeto, atendiendo a qué recursos las componen y qué ficheros contienen estos últimos. El panel central de la herramienta, el encargado de mostrar el contenido de los objetos, quedará dividido en dos marcos: un menú izquierdo que muestra las organizaciones y los recursos asociados a cada una de éstas, y otro que mostrará o el logotipo de la herramienta (si el objeto acaba de ser abierto o no tiene recursos), o el contenido de uno de los recursos del objeto de aprendizaje.

2. Edición de un objeto de aprendizaje

La edición será capaz de permitir al usuario de la herramienta añadir nuevos hijos, modificar los atributos y eliminar elementos del objeto de aprendizaje. Estas funciones se gestionarán mediante el árbol en el que se muestra la estructura del manifiesto asociado al objeto de aprendizaje con el que se está trabajando.

Se pueden diferenciar dos tipos de metadatos, aquellos que pueden tener hijos (compuestos por otros metadatos, y considerados como *nodos padre*) y aquellos que no pueden tener hijos (no están compuestos por otros metadatos, y lo que contienen es un valor determinado. Se les considera *nodos hijo* u *hoja*).

Así, cada vez que el usuario pincha con el botón derecho del ratón sobre un elemento del árbol aparecerá un menú en el que se le darán distintas posibilidades atendiendo al tipo de nodo sobre el que se ha interactuado. Concretamente en los *nodos padre* se puede: insertar nodo, eliminar nodo, añadir atributos y modificar atributos; y en los *nodos hijo* se puede: insertar información, modificar información, añadir atributos y modificar atributos. Cada una de las acciones que selecciona el usuario, se ven reflejadas inmediatamente en el árbol.

2.1. Inserción de nuevos elementos

Como ya se ha comentado, la herramienta permitirá la inserción de nuevos elementos. La gestión de esta funcionalidad se realizará mediante un menú emergente que aparecerá al pinchar el usuario con el botón derecho sobre uno de los nodos del árbol de edición de contenidos.

El contenido del menú emergente en el que aparecerán las diferentes acciones que se pueden realizar sobre el nodo seleccionado será creado en tiempo de ejecución, atendiendo al contenido del nodo sobre el que se ha interactuado y a la estructura marcada por la gramática del objeto de aprendizaje.

La posibilidad de insertar un elemento, por tanto, sólo se encontrará en el menú emergente si el nodo sobre el que hemos pulsado es un nodo padre, y los elementos que se podrán insertar estarán limitados a los elementos que puedan ser hijos del nodo seleccionado, según indique la gramática.

Por ejemplo, si pinchásemos con el botón derecho sobre el nodo *organizations,* debería aparecernos la posibilidad de añadir tan sólo un nodo de tipo *organization*, que es el único hijo que admite el anterior.

2.2. Eliminación de un elemento

En el menú emergente, resultado de pulsar con el botón derecho sobre un nodo del árbol, siempre deberá aparecer la posibilidad de eliminar el elemento sobre el que se pinchó.

2.3. Modificación de atributos

La gestión de esta funcionalidad de la herramienta se realizará en el panel inferior de la ventana principal de la herramienta de autoría.

Todos los nodos sobre los que se interactúe pinchando con el botón derecho, sean del tipo que sean, ofrecerán al usuario la posibilidad de modificar los atributos de que constan, si es que tienen alguno. Se mostrará en el menú emergente asociado a la pulsación sobre el árbol una opción *modificar.* Al seleccionar esta opción, en el panel inferior de la herramienta aparecerán todos los posibles atributos del nodo elegido y un campo de texto con el contenido actual, si lo tienen. El usuario podrá entonces modificar el contenido de los atributos alterando el contenido de los campos de texto.

3. Guardar el manifiesto

El sistema deberá permitir guardar los cambios realizados durante la edición del objeto de aprendizaje.

Para ello la herramienta tendrá que ser capaz de escribir el contenido del árbol de edición de contenidos en el fichero imsmanifest.xml, que almacena la estructura del objeto de aprendizaje.

4. Guardar un objeto de aprendizaje en forma comprimida

Se solicitará al usuario mediante una ventana de navegación la ubicación y el nombre del fichero .zip donde se guardará el objeto de aprendizaje.

El paquete que se creará dentro del fichero zip respetará las especificaciones que formen el contexto sobre el que fue creado o editado dicho objeto. De este modo, el paquete mínimo contendrá los ficheros de manifiesto (imsmanifest.xml), esquema de manifiesto (imscp_v1p1.xsd) y esquema de metadatos (imsmd_v1p2p2.xsd). Además de estos contenidos mínimos, cada fichero deberá contener los ficheros a los que apuntan los recursos del mismo, con el objetivo de que el objeto sea autosuficiente, es decir, contenga en su interior toda su información.

5. Creación de un nuevo objeto de aprendizaje

La herramienta permitirá, además de la edición de objetos de aprendizaje creados anteriormente, la creación desde cero de objetos de aprendizaje. Es decir, el usuario podrá crear objetos de aprendizaje a partir de una plantilla mínima, la que impone la especificación de IMS Content Packaging. A partir de ese objeto mínimo el usuario podrá ir añadiendo elementos y atributos hasta lograr el objeto deseado.

Al pulsar sobre el botón de crear nuevo objeto de aprendizaje se solicitará al usuario que elija la ubicación del contenido del nuevo objeto de aprendizaje, creando en el directorio elegido por el usuario los tres ficheros básicos de todo objeto de aprendizaje: imsmanifest.xml, imscp_v1p1.xsd e imsmd_v1p2p2.xsd. El fichero imsmanifest.xml contendrá la estructura mínima obligatoria o plantilla mínima.

La plantilla mínima que se generará automáticamente al seleccionar la opción de *crear nuevo objeto de aprendizaje,* constará de los elementos obligatorios requeridos por la especificación sobre la que basará su estructura. Además de los elementos obligatorios en la gramática, se generará la raíz de los metadatos del objeto, el elemento *lom.* La plantilla básica, representada en forma de árbol, será la siguiente:

manifest
|--- *metadata*
|--- *organizations*
|--- *resources*

A partir de este objeto mínimo, que se mostrará en el árbol de edición de contenidos tras ser generado, el usuario podrá generar el objeto de aprendizaje que desee, utilizando el resto de opciones de la herramienta de autoría.

6. Ayuda

6.1. Manual de usuario

Se ofrecerá al usuario la posibilidad de consultar un manual de ayuda sobre el manejo de la herramienta, en el que se recojan todas y cada una de las posibilidades de la misma y qué secuencia de pasos sería necesaria para realizarla.

También se explicarán en el manual los usos de los botones de la barra de herramientas, menús y secuencias de escape para el acceso rápido a cada una de las opciones de la aplicación.

6.2. Información corporativa

La herramienta dispondrá de acceso a un panel de información corporativa, detalles de la versión de la herramienta, desarrolladores, etc.

6.3. Información de elemento

En el panel inferior de la parte derecha de la pantalla principal de la herramienta, se mostrará dinámicamente información sobre cada uno de los nodos del árbol sobre los que se pulse con el ratón, con el objetivo de facilitar la creación de contenidos a usuarios noveles en el uso de especificaciones e-learning.

Módulo 2: Módulo de composición

1. Mostrar objetos de aprendizaje

Esta opción permite al usuario elegir el objeto de aprendizaje que quiere abrir para su posible posterior composición con otro objeto de aprendizaje. Esta opción abre los objetos de aprendizaje de forma que sólo muestra el nodo organizations de los mismos. Esto es así, ya que sólo es necesario para el usuario conocer este elemento del objeto de aprendizaje para su posible composición con otro.

De la misma forma que ocurría al abrir objetos de aprendizaje (explicado en el punto 1 del módulo 1), se muestra al usuario una ventana de navegación, en la que deberá elegir el objeto que desea mostrar para su composición. El objeto que elija deberá estar en formato zip y contener en su directorio raíz los ficheros correspondientes a su manifiesto, esquema de manifiesto y esquema de metadatos.

Una vez seleccionado el objeto por el usuario, primeramente se comprueba que este objeto no está mostrado actualmente en el árbol de composición o que no existe ya en el árbol un objeto con el mismo nombre (aunque no se trate del mismo objeto). En cualquiera de estos casos se informará al usuario de que, por estos motivos, no se podrá mostrar el objeto seleccionado.

Si esto no ocurriera, se realizará la validación del manifiesto del objeto de aprendizaje (imsmanifest.xml) frente a su esquema de manifiesto (imscp_v1p1.xsd) y a su esquema de metadatos (imsmd_v1p2p2.xsd).

Si el objeto de aprendizaje supera correctamente la validación, se procederá a la descompresión de los archivos del mismo.

Los archivos descomprimidos serán guardados en una carpeta temporal creada expresamente para este fin, de forma dinámica, y donde se guardarán todos los cambios que el usuario realice durante la composición.

A partir de este objeto de aprendizaje se creará un objeto de composición. Este tipo de objetos contienen únicamente la información del elemento organizations del objeto de aprendizaje que representa, extraída del documento

imsmanifest.xml. Éste será el objeto que se mostrará en el árbol de composición en el panel izquierdo de la aplicación.

Si el objeto no superara la validación, se mostraría un mensaje de error y no se mostraría dicho objeto en el árbol de composición.

El árbol de composición estará formado por tantos objetos de composición como objetos de aprendizaje muestre el usuario excepto los que va componiendo como hijos, que irán desapareciendo del mismo.

2. Composición en paralelo

Esta opción permite al usuario realizar la composición en paralelo de dos objetos de aprendizaje. Únicamente estará habilitada si se han mostrado al menos dos objetos en el árbol de composición.

La composición en paralelo consiste en componer dos objetos de forma que el objeto que actúa como contenido sea un hijo directo de la raíz de una organización del objeto que actúa como contenedor.

Por ello, al clickear sobre esta opción se abre un formulario que muestra los nombres de todos los objetos de aprendizaje abiertos previamente, cada uno con sus organizaciones. El usuario elegirá una única organización, y el objeto al que pertenezca será el objeto continente.

Una vez que el usuario ha aceptado este formulario, se abre el mismo formulario en el que se muestran los nombres de todos los objetos excepto el elegido como objeto continente en el formulario anterior (mostrando de nuevo las organizaciones de cada uno de los objetos). El usuario debe seleccionar otro objeto, que actuará como objeto contenido.

Al aceptar en este segundo formulario se lleva a cabo la composición, que se realiza de la siguiente forma:

a) Se valida el nuevo objeto de aprendizaje compuesto. Si el objeto compuesto no fuera validado se mostraría un mensaje de error y el objeto no sería compuesto.

b) Se actualiza el árbol mostrando el nuevo objeto y eliminando del mismo el objeto elegido como hijo.

c) Si quedan más de dos objetos abiertos en el árbol de navegación se mantienen habilitadas las opciones de composición en paralelo y en profundidad, en caso contrario se deshabilitan.

El paquete que se creará dentro del fichero zip respetará las especificaciones que formen el contexto sobre el que fue creado o editado el objeto que actuó como padre en la composición. De este modo, el paquete mínimo contendrá los ficheros de manifiesto (imsmanifest.xml), esquema de manifiesto (imscp_v1p1.xsd) y esquema de metadatos (imsmd_v1p2p2.xsd). Además de estos contenidos mínimos, cada fichero deberá contener los ficheros a los que apuntan los recursos del mismo, con el objetivo de que el objeto sea autosuficiente, es decir, contenga en su interior toda su información.

3. Composición en profundidad

Esta opción permite al usuario realizar la composición en profundidad de dos objetos de aprendizaje. Únicamente estará habilitada si se han mostrado al menos dos objetos en el árbol de composición.

La composición en profundidad consiste en componer dos objetos de forma que el objeto que actúa como contenido va a ser un hijo de algún hijo interno de la raíz de una organización.

Por ello, al elegir esta opción se abre un formulario que muestra los nombres de todos los objetos de aprendizaje mostrados previamente, cada uno con todos los ítems de sus organizaciones. El usuario elegirá un único ítem, y el objeto al que pertenezca el mismo será el objeto continente.

Una vez que el usuario ha aceptado este formulario, se abre un segundo formulario en el que se muestran los nombres de todos los objetos excepto el elegido como objeto continente en el formulario anterior (mostrando en el mismo las organizaciones de cada uno de los objetos). El usuario debe seleccionar una de estas organizaciones, y el objeto al que pertenezca actuará como objeto contenido.

Al aceptar en este segundo formulario se lleva a cabo la composición en profundidad, que se realiza de la misma forma que se explicó anteriormente para la composición en paralelo.

Al igual que en la composición en paralelo, el paquete que se creará dentro del fichero zip respetará las especificaciones que formen el contexto sobre el que fue creado o editado el objeto que actuó como padre en la composición. De este modo, el paquete mínimo contendrá los ficheros de manifiesto (imsmanifest.xml), esquema de manifiesto (imscp_v1p1.xsd) y esquema de metadatos (imsmd_v1p2p2.xsd). Además de estos contenidos mínimos, cada fichero deberá contener los ficheros a los que apuntan los recursos del mismo, con el objetivo de que el objeto sea autosuficiente, es decir, contenga en su interior toda su información.

4. Guardar objetos compuestos en forma comprimida

Al pulsar esta opción el sistema comprueba que previamente se realizó al menos una composición entre dos objetos. Sino, se mostrará un mensaje de error advirtiendo al usuario que no se realizó ninguna composición y, por tanto, no hay ningún objeto compuesto para guardar.

Si se realizó alguna composición, se mostrará al usuario una ventana en la que deberá seleccionar el objeto compuesto que desea guardar comprimido. Una vez seleccionado el objeto y aceptada la ventana, se muestra un mensaje informando al usuario que el objeto ha sido comprimido satisfactoriamente en el destino dado, y se le recuerda esta ruta destino.

Si existe más de un objeto compuesto, se pregunta al usuario si desea guardar otro, si la respuesta es afirmativa, se mostrará de nuevo la ventana de navegación repitiendo los pasos anteriores.

5. Guardar todo

Al pulsar esta opción el sistema comprueba que previamente se realizó al menos una composición entre dos objetos y que alguno de ellos no ha sido guardado aún.

Si esto no se cumpliera, se mostraría un mensaje de error advirtiendo al usuario que actualmente no existe ningún objeto compuesto no guardado, y no se realizará ninguna acción.

Si se realizó alguna composición cuyo objeto compuesto aún no ha sido guardado por el usuario en formato zip, se le mostrará una ventana de navegación informando del objeto compuesto que se va a guardar, y en la que deberá elegir el destino donde se comprimirá este objeto. Una vez aceptada esta ventana, el objeto será comprimido y se mostrará un mensaje informando al usuario de la ruta en la que éste ha sido guardado de forma satisfactoria.

Esto se repetirá para cada uno de los objetos compuestos creados hasta el momento y que no hayan sido guardados.

6. Salir de la composición

La selección de salir de la composición informa al usuario de que al salir, los cambios que realizó se perderán si no fueron guardados previamente. Se le pregunta si está seguro que desea salir y si su respuesta es afirmativa la pantalla de la aplicación se vaciará, el árbol de composición desaparecerá de forma que los paneles quedarán vacíos y la herramienta aparecerá tal y como estaba al comienzo.

El sistema eliminará las carpetas temporales que se crearon al comenzar la composición.

Si la respuesta es negativa por parte del usuario, entonces no se realizará ninguna acción.

Módulo 3: Módulo de evaluación de calidad

1. Evaluación manual de la calidad de objetos de aprendizaje

La herramienta deberá permitir realizar la evaluación de objetos de aprendizaje desde un punto de vista general, es decir, desde una visión superficial, la que ofrece la parte de los metadatos generales del objeto de aprendizaje.

El sistema dará la posibilidad al usuario de realizar una evaluación manual, indicando este último el valor de cada uno de los elementos que se tendrán en cuenta a la hora de llevar a cabo la evaluación general del objeto de aprendizaje.

Para llevar a cabo esta funcionalidad el sistema mapeará el contenido del objeto de aprendizaje abierto en el momento en el que se pincha sobre el botón de evaluación manual, y lo mostrará en una ventana de evaluación, en la que el usuario podrá analizar el contenido de cada una de las etiquetas que formarán parte de la evaluación global del objeto. Un vez haya analizado en contenido, el usuario deberá seleccionar una valoración para cada una de las etiquetas que aparezcan en la ventana de evaluación. Con todos estos valores, el sistema realizará estimación sobre la evaluación global del objeto, basada en una métrica previamente establecida, que se mostrará al usuario. Si el usuario aceptase esta evaluación de calidad, la estructura del objeto se guardaría en la base de datos del sistema, junto con su evaluación.

Los posibles valores de evaluación de una etiqueta del objeto serán:

MUY BUENO
BUENO
REGULAR
MALO

MUY MALO

La información de la base de datos servirá posteriormente de base de casos del sistema para la realización de la evaluación automática de objetos de aprendizaje basada en CBR (Case Based Reasoning).

2. Clasificación automática de objetos de aprendizaje

Esta funcionalidad dotará al sistema de inteligencia, y permitirá evaluar de manera automática, sin esfuerzos por parte del usuario, la calidad de un objeto de aprendizaje, siguiendo unos parámetros generales.

Pinchando sobre el botón de evaluación automática, el sistema proporcionará al usuario una estimación, basada en el resultado de una comparación CBR con el resto de objetos existentes en la base de datos. Se presentará esta estimación al usuario, el cual podrá aceptarla, con lo que estimación y objeto serán guardados en la base de datos (aprendizaje); o cambiarla manualmente, asociando a cada etiqueta relevante para la evaluación una valoración entre *MUY BUENO* y *MUY MALO*, como ya se explicó en el apartado anterior.

Esta evaluación se realizará basándose en un algoritmo CBR (*Case Based Reasoning* o Razonamiento Basado en Casos). Los resultados obtenidos mediante este método dependerán, por tanto, del tamaño de la base de casos del sistema y la calidad de las evaluaciones que allí se encuentren almacenadas.

Inicialmente, la base de casos del sistema no contendrá ningún caso, por lo que será el usuario el encargado de aumentar y cuidar las evaluaciones que se guarden para poder dotar al CBR de una base de casos correcta sobre la que operar.

3.3.2 Servicios potenciales (requisitos emocionantes)

1. Consultas sobre una base de datos que contiene información de objetos de aprendizaje.

Esta funcionalidad tratará de dotar al sistema de un elemento de acceso rápido a los objetos de aprendizaje guardados en la base de datos. Se proporcionará, de esta forma, al usuario una interfaz sobre la cual podrá configurar los parámetros de cada búsqueda. Si el usuario no rellenase ningún parámetro se mostrarán todos los objetos almacenados en la base de datos en ese instante.

Los parámetros de búsqueda sobre los que se permitirá realizar búsquedas, serán etiquetas de metadatos del objeto de aprendizaje o estarán relacionados con la evaluación asociada a cada objeto de la base de datos.

Además, esta funcionalidad enlazará con la herramienta de autoría de objetos ya que se podrá acceder a la edición de cualquiera de los objetos buscados. Si el usuario pincha en el objeto deseado, se abrirá una ventana que solicitará la ubicación donde se quiere descomprimir el objeto de aprendizaje, para proceder después de esto a la validación, apertura y visualización del objeto seleccionado.

2. Creación de un espacio Web en el que promocionar la herramienta

Uno de los medios que más posibilidades ofrece a la hora de promocionar una herramienta, y a menos coste, es Internet. Por tanto, sería interesante la creación de un espacio Web sencillo en el que exponer la herramienta a la comunidad que está trabajando en este campo en pleno auge que es el e-learning.

La Web a crear debería ofrecer a sus usuarios la opción de descargar la herramienta: código fuente, instalador, manuales de usuario e instalación, etc.

Dando de alta la Web creada entonces en los distintos buscadores, se podría conseguir que llegara a la comunidad de desarrollo en el campo del e-learning y su dictamen sobre la herramienta sería de gran utilidad en posibles futuras ampliaciones de la misma.

3.4 Restricciones y limitaciones – Requisitos no funcionales

3.4.1 Entorno operativo

Requisitos Hardware mínimos

- Portátil con pantalla TFT 14.1" o mayor con área de pantalla >= 1024 x 768
- Monitor 15" con área de pantalla >= 1024 x 768
- Espacio libre en disco: 90 MB
- Pentium® MMX® 233 o AMD® 500
- 128 MB de RAM
- Lector de CD-ROM 16x, ratón y teclado

Requisitos Hardware óptimos

- Portátil con pantalla TFT 14.1" con área de pantalla = 1024 x 768
- Monitor 17" con área de pantalla = 1024 x 768
- Espacio libre en disco: 100 MB
- Pentium® III 700 o AMD® 1250
- 512 MB de RAM
- Lector de CD-ROM 32x, ratón y teclado

Requisitos Software mínimos

- Microsoft® Windows® 98 / 2000 / XP
- Máquina virtual de Java con JDK 1.4.2 o posterior
- Navegador Microsoft® Internet Explorer 5 o posterior

Requisitos Software óptimos

- Microsoft® Windows® XP
- Máquina virtual de Java con JDK 1.5
- Navegador Microsoft® Internet Explorer 6

3.4.2 Interfaces externas

El sistema tiene relación con diversos elementos externos, los cuales proveen de servicios indispensables para el buen funcionamiento de la aplicación.

Sistema operativo:

La principal interacción entre la herramienta y el sistema operativo se producirá en el mostrado del contenido de los objetos de aprendizaje, la cual, debido a las restricciones que impone el API de Java a la hora de soportar diferentes formatos de documentos, diferenciará los ficheros a mostrar en dos grupos: los que se pueden mostrar directamente en la herramienta, y los que deben mostrarse a través de navegador ya que el sistema no soporta su formato. Cuando se detecte que el formato de uno de los ficheros de los que se compone el objeto de aprendizaje no es soportado se lanzará el navegador Microsoft® Internet Explorer, y el fichero será mostrado allí.

Periféricos:

El ratón y el teclado serán los periféricos operativos con los cuales podremos movernos por la herramienta y realizar todos los accesos a sus funciones.

Máquina virtual de JAVA:

Internamente, el código de la aplicación será interpretado por la máquina virtual de JAVA. Debido a algunas de las librerías y funciones avanzadas es estrictamente necesario que el usuario disponga de una máquina virtual con versión de JDK 1.4.2 o superior.

Servidor de bases de datos MySQL®:

El módulo de evaluación de calidad de objetos de aprendizaje basa su funcionamiento en una base de casos, implementada en una base de datos, que estará albergada en un servidor de bases de datos MySQL®.

3.4.3 Documentación de usuario y otras asistencias

Para facilitar la distribución del software se realizarán diversos tipos de manuales destinados al usuario:

1. Manual de usuario (Tutorial de manejo de la herramienta)

Se realizará un manual de usuario muy detallado en el que se incluirán capturas de la aplicación demostrando de manera visual las posibilidades que la herramienta ofrece al usuario, y comentando específicamente, para cada una de éstas posibilidades, los pasos que serían necesarios para llevarlas a cabo.

Otro aspecto que sería interesante incluir en el manual de usuario sería la descripción de cada uno de los botones de la barra de la herramienta, en la que el usuario obtenga una mayor ayuda que la que se pueda introducir en la herramienta mediante textos de ayuda emergentes.

Como este manual pretende ser una guía útil para el usuario, se incluirá en el mismo un ejemplo detallado de la creación de un objeto de aprendizaje desde la plantilla básica de la herramienta y pasando por todas las funcionalidades disponibles tanto en la herramienta de autoría como en los módulos de composición y de evaluación de calidad de objetos de aprendizaje.

Este manual deberá incluir además información sobre todas las especificaciones utilizadas en la creación de objetos de aprendizaje, así como enlaces en los que encontrar más información sobre las mismas.

2. Instalador de la aplicación

Como complemento fundamental para facilitar la distribución se creará un instalador de la herramienta que se encargará, no sólo de instalar la propia aplicación, sino de configurar todas y cada una de las librerías que ella utilice.

Además el instalador deberá ser capaz de ofrecer la posibilidad de instalar el resto de aplicaciones que son estrictamente necesarias para su correcto funcionamiento. Así, el instalador se encargará de facilitar una máquina virtual o permitir al usuario que elija una ya instalada en el equipo; instalar correctamente MySQL® Server, servidor de bases de datos de la aplicación; y crear la base de datos en el servidor y crear y configurar las tablas necesarias para que la aplicación funcione correctamente.

De esta forma, el usuario no será necesario que el usuario tenga ningún conocimiento sobre bases de datos, ampliando así los horizontes de propagación de la herramienta.

3. Manual de instalación de la aplicación

Aunque el manejo del instalador es de lo más simple, se proporcionará también al usuario de la aplicación un manual de instalación en el que se indicarán de forma minuciosa los pasos a seguir para la correcta instalación de la herramienta. Se incluirán también imágenes de cada uno de los pasos de la instalación, requisitos mínimos del sistema, etc.

3.4.4 Requerimientos de desarrollo

El equipo de desarrollo de la herramienta estará formado por tres integrantes, dirigidos por el profesor y director del proyecto, que se encargará de establecer las funcionalidades y dictaminar sobre la calidad del software desarrollado, actuando como actuaría un cliente de software real.

Las principales herramientas a destacar en el desarrollo del proyecto LOMEditor son las siguientes:

- Borland® JBuilder Enterprise 2005 / JCreator® v3.0 / NetBeans® IDE

Debido a las facilidades a la hora de realizar interfaces de usuario profesionales se intentará utilizar preferentemente Borland JBuilder. No obstante, debido a que este es un programa que consume muchos recursos, y las instalaciones de que disponemos en la Universidad para trabajar no lo soportan se trabajará, en segunda opción, con JCreator v3.0. Este está mucho más limitado en sus posibilidades, pero está disponible en laboratorios, ya que es software libre y no consume demasiados recursos.

NetBeans se comenzó utilizando, pero fue descartado más tarde porque no daba la posibilidad de alterar las interfaces de usuario de manera manual, sólo era posible mediante el diseñador de la propia herramienta, que a su vez no permitía realizar todas las acciones que se nos antojaban necesarias.

- UMLPad / dia

UMLPad se utilizará para el desarrollo de los diagramas de clases y secuencia que se encuentran en el apéndice A.

dia es una herramienta vectorial que será utilizada para la realización de diagramas de bases de datos, tanto relacionales como de entidad-relación (E-R).

- *MySQL® Server 4.0*

Se utilizará como servidor de bases de datos. Es imprescindible para que el módulo de evaluación de objetos de aprendizaje funcione correctamente, ya que el CBR que utiliza este módulo implementará su base de casos en una base de datos relacional MySQL.

- *MySQL® Control Center*

Esta aplicación, que MySQL ofrece libremente a sus usuarios, es utilizada principalmente como interfaz amigable sobre la base de datos. El usuario de LOMEditor no requerirá de este programa, ya que el trabajo sobre la base de datos es transparente para el. Sin embargo, el desarrollador del sistema encuentra en MySQL Control Center una herramienta muy útil para su trabajo. Será, por tanto, utilizada principalmente en labores de desarrollo y pruebas sobre la base de datos.

- *Install Anywhere® 6 Enterprise*

La herramienta LOMEditor está compuesta por diferentes el elementos que forman un sistema demasiado complejo como para que un usuario no especializado pueda acceder cómodamente al mismo. Con este fin se utiliza Install Anywhere® 6 Enterprise, una aplicación que permite generar un instalador de fácil ejecución para LOMEditor. De esta manera contaremos con un sistema de instalación realmente sencillo, mediante el que cualquier usuario podrá hacer uso de la aplicación.

- *Macromedia® Dreamweaver®*

LOMEditor no solo es una herramienta de creación de objetos de aprendizaje, sino que permite además la visualización de los mismos. Dado el importante carácter audiovisual de los recursos que componen el objeto, se recurre a la creación de plantillas y facilidades en lenguajes de marcado, como páginas Web, etc. Macromedia Dreamweaver es la aplicación elegida para la creación de todo ello.

- *Macromedia® Fireworks®*

Esta aplicación de Macromedia permite la creación de todo tipo de imágenes y diseños gráficos. La imagen de la herramienta LOMEditor ha sido ideada y creada gracias a ella. Cabe destacar la sencillez de este software, que ofrece gran cantidad de servicios dentro del ámbito del diseño gráfico.

- *ExamDiff® Pro*

Con el fin de facilitar la completa tarea de integración de código se recurre a la aplicación ExamDiff Pro. Esta herramienta permite comparar diferentes ficheros, resaltando sus similitudes y diferencias, lo cual favorece a agilizar de forma sensible el proceso de integración en cada entrega de prototipos.

- *CuteFTP® Professional*

Una aplicación muy fiable y de gran prestigio dentro del sector. Simplifica y optimiza la transferencia de ficheros. Entre otros usos, ha servido para incorporar diversos ficheros al portal Web donde se presenta LOMEditor.

- *UltraEdit®*

Este excelente editor de textos es utilizado por los desarrolladores en su trabajo con todo tipo de ficheros. Su facilidad de resaltar múltiples códigos fuente hace de UltraEdit una cómoda herramienta para la revisión de fuentes Java, así como la creación o modificación de manifiestos en xml o gramáticas xsd, necesarios en las pruebas de funcionalidad.

Resulta además muy recomendable su utilización en labores donde prima la rapidez, pues se trata de un software verdaderamente ligero, y muy comprometido en este sentido.

- *XMLSpy® 5*

Para un trabajo más avanzado sobre ficheros XML es más recomendable y apropiado utilizar software dedicado a este campo. XMLSpy ha sido el elegido para ayudarnos en estas labores.

- *Microsoft® Word®*

Este popular editor de texto colabora en la creación de documentos y manuales que supondrán el soporte del equipo de desarrollo.

3.5 Métodos de prueba

3.5.1 Tipos de pruebas

LOMEditor, al igual que otras aplicaciones de similares características, en cuanto a extensión y complejidad, requiere de un desarrollo organizado en equipo, dividiendo el trabajo en varias entregas o fases. Este planteamiento es muy positivo, y presenta diversas ventajas que ya hemos podido analizar, pero también conlleva riesgos a tener en cuenta, uno de ellos, el aumento potencial de errores.

Para solucionar este problema se debe definir con mucho cuidado un plan de pruebas en cada entrega. Esta entrega incluirá las pruebas Personales del desarrollador, y las pruebas de Certificación, una vez integrados los desarrollos de todos los componentes del equipo. Finalmente debemos realizar una batería de pruebas sobre el producto finalizado.

Atendiendo al periodo de realización de las pruebas, diferenciamos por tanto estos tres casos:

1. *Pruebas Personales*

Una vez completado el desarrollo planificado a nivel personal por cada componente del equipo, este debe probar la nueva funcionalidad que ha implementado, asegurándose de su correcto comportamiento. Asimismo, debe comprobar que toda la herramienta se mantiene estable, y los servicios ya ofrecidos no se han visto afectados negativamente.

Cada elemento del grupo debe aportar los resultados de sus pruebas, pues en caso de ser satisfactorias, se validará la entrega, mientras que si se produce algún problema, deberá actuarse en consecuencia.

Es recomendable reflejar el plan de pruebas por escrito, con el fin de aplicarlo de la forma más rigurosa posible.

2. *Pruebas de Certificación*

Una vez pasadas las pruebas personales por cada miembro del equipo de desarrollo, se realizará la integración de cada componente, en busca de cerrar la entrega de un nuevo prototipo.

En esta fase del desarrollo deben realizarse las pruebas con más cuidado que en ningún otro caso, ya que el riesgo de errores es elevado al manejar código de varias personas.

En primer lugar debe comprobarse la nueva funcionalidad que cada desarrollador ha aportado. Para agilizar este proceso es conveniente que cada integrante del equipo aporte su plan de pruebas para que nada se pase por alto. Resulta positivo que estas pruebas las realice alguien, que aun perteneciendo al equipo, no haya desarrollado la parte probada en si, pues colabora a descubrir posibles conflictos que pasaron desapercibidos.

Una vez probada la funcionalidad más recientemente incorporada, se debe realizar una completa batería de pruebas con el fin de cerciorarse del correcto comportamiento del resto de la herramienta, pues, si importante es que funcionen los nuevos casos incorporados, resulta primordial no generar errores en partes anteriores ya catalogadas de estables.

De esta forma se consigue un nuevo prototipo estable y fiable a partir del cual continuar el desarrollo.

3. *Pruebas de Certificación finales*

Antes de cerrar definitivamente la primera versión completa del producto deben repetirse de nuevo todas las principales pruebas. Hay que tener en cuenta que no habrá otra oportunidad de realizar reparos de esta forma, y cualquier problema conllevará la creación de parches o nuevas versiones solventando el error.

Es conveniente que en estas pruebas finales esté presente el cliente o algún usuario potencial del producto, pues aportará puntos de vista muy útiles para terminar de depurar nuestro desarrollo.

Una vez pasadas estas últimas pruebas se podrá asegurar que se entrega un producto estable, fiable y completo, que cumple con las expectativas generadas.

Profundizando un poco más en los tipos de pruebas a realizar sobre LOMEditor, distinguiremos estos tres tipos de pruebas:

1. *Pruebas específicas*

Como su nombre indica, estas primeras pruebas tienen el fin de comprobar el correcto funcionamiento de una parte específica de la herramienta. Por supuesto, se tratará del último desarrollo realizado sobre la misma.

Estas pruebas cobran especial importancia en la fase de pruebas personales, ya que es responsabilidad de cada desarrollador en particular que la nueva funcionalidad cumpla correctamente con lo esperado. Siendo además él mismo el mayor conocedor de estos nuevos servicios, debe asegurarse de que actúan conforme a lo especificado y no presentan conflictos ni errores.

En la fase de pruebas a realizar tras la integración de los diversos desarrollos, no obstante, se acabará de certificar el buen comportamiento de la herramienta.

2. Pruebas de regresión

Estas importantes pruebas buscan asegurar que la herramienta no pierde estabilidad ante los nuevos desarrollos, ni se generan conflictos o errores.

Tras pasar las pruebas específicas y dar el visto bueno a la nueva funcionalidad, es el turno de analizar completamente la herramienta. Estas pruebas no alcanzan la profundidad de las anteriores, pero si abarcan los principales aspectos del producto.

En la fase de certificación tendrán mucha importancia, pues será entonces cuando esté actuando conjuntamente toda la nueva funcionalidad de LOMEditor, lo cual aumenta el riesgo de fallos o conflictos.

Tras pasar estas pruebas, podemos hablar de un sistema completamente estable en condiciones normales.

3. Pruebas de carga

Si bien es cierto que esta clase de pruebas son más apropiadas en sistemas que van a ser utilizados por un gran número de personas de forma concurrente, no hemos querido pasar sin realizar algunas comprobaciones mínimas.

Dado que el acceso a LOMEditor es a nivel local, los problemas derivados de carga de proceso están más localizados en posibles conflictos tras su utilización continuada.

Para ello se someterá LOMEditor a un intenso trabajo, buscando ver su comportamiento.

Estas pruebas, que como anteriormente hemos mencionado, son de menor importancia y se realizarán casi exclusivamente para prototipos con trabajo de bases de datos, y producto final.

3.5.2 Método de actuación en caso de pruebas no superadas

Es probable, y resulta muy recomendable a la hora de garantizar la corrección del producto, que ocasionalmente se produzcan pruebas no superadas, tanto a nivel de pruebas personales, como de certificación.

La actuación a seguir será la siguiente:

En caso de localizarse algún problema en la aplicación durante el período de pruebas personales, el desarrollador tiene la responsabilidad de subsanar el error, realizando de nuevo todo el proceso de comprobación que estaba planificado, aunque con ello pueda retrasarse la entrega, y por tanto, la integración y cierre de un nuevo prototipo. Estos casos están previstos en la planificación de cada entrega y no suponen un problema grave, sino más bien vienen a confirmar la seguridad del producto.

Si por el contrario el problema se produce durante las pruebas de certificación, se pasará a trabajar sobre la aplicación ya integrada, en caso de que el conflicto no sea muy grave. El encargado de repararlo será un desarrollador designado por el grupo. Si el conflicto es importante, y hace totalmente inestable la aplicación, deberá estudiarse qué nuevo módulo ocasiona estos fallos, encargándose su desarrollador de volver a implementarlo y probarlo personalmente, para después integrarlo de nuevo y pasar las pruebas.

Se vuelve a repetir que es recomendable pasar por estos procesos, ya que contribuyen a garantizar el buen funcionamiento del producto. Todos los errores localizados durante el plan de pruebas serán errores que el cliente no sufrirá.

3.5.3 Resultados esperados

Con este proceso de pruebas y certificación se busca principalmente poder ofrecer al cliente un producto que cumpla sus expectativas y garantice su funcionamiento.

Más concretamente buscamos construir una herramienta altamente estable, que ofrezca un elevado nivel de fiabilidad al usuario en cualquier situación de uso en la que éste se pueda encontrar.

A nivel de diseño, tras una serie de prototipos, el objetivo es que LOMEditor alcance a satisfacer los deseos del cliente en este aspecto, sin que por ello su funcionalidad o robustez se vean en absoluto afectadas.

A nivel de comportamiento se busca y espera obtener un producto a la altura de sus competidores en este sector, superando incluso a estos últimos en cuanto a funcionalidad y servicios.

En resumen, este proceso tiene por misión hacer realidad la herramienta planificada.

SECCIÓN 4ª

CASOS DE USO DEL SISTEMA

4.1 Módulo 1: Herramienta de autoría

4.1.1 Caso de uso "Abrir objeto de aprendizaje" (LOME01)

Objetivo en concreto	Apertura manual de un objeto de aprendizaje que se encuentra guardado en formato zip. Descomprimiremos el fichero y mostraremos el objeto en la herramienta.	
Entradas	Objeto de aprendizaje en formato zip, directorio destino.	
Precondiciones	El fichero zip debe contener un objeto de aprendizaje correcto y la carpeta destino debe existir.	
Salidas	Ficheros descomprimidos en la carpeta destino.	
Poscondición si éxito **Poscondición si fallo**	El objeto de aprendizaje se descomprime en el directorio destino y se muestra su contenido en la herramienta. Se muestra al usuario una ventana de error.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede al Menú Principal, Archivo, Abrir, Abrir Objeto de Aprendizaje ó al botón Abrir de la barra de herramientas.
	2	El sistema mostrará un cuadro de diálogo para navegar por el equipo del usuario.
	3	Seleccionamos en el cuadro de diálogo el fichero .zip que contiene el objeto de aprendizaje que deseamos abrir.
	4	Seleccionamos la carpeta destino donde vamos a descomprimir el contenido del objeto de aprendizaje.
	5	El sistema analiza el objeto de aprendizaje, comprobando su corrección. Si lo valida, se muestra.
	6	El sistema mostrará un árbol de directorios donde quedará reflejado el objeto de aprendizaje, facilitando de esta forma su edición.
Secuencia alternativa	**Paso**	**Acción**
	6	Si el objeto de aprendizaje no es válido, el sistema no mostrará el árbol de representación, sino que alertará de lo ocurrido mediante un mensaje de error.

4.1.2 Caso de uso "Nuevo objeto de Aprendizaje" (LOME02)

Objetivo en concreto	Apertura automática de un objeto de aprendizaje mínimo como punto de partida para la creación de un nuevo objeto de aprendizaje.	
Entradas	Objeto de aprendizaje por defecto. Carpeta destino para el nuevo objeto de aprendizaje.	
Precondiciones	El objeto de aprendizaje por defecto y la carpeta destino deben existir.	
Salidas	Ficheros descomprimidos en la carpeta destino.	
Poscondición si éxito **Poscondición si fallo**	Se crea un nuevo objeto de aprendizaje en el directorio destino indicado. Se muestra al usuario una ventana de error.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede al Menú Principal, Archivo, Nuevo, Nuevo Objeto de Aprendizaje ó al botón Nuevo de la barra de herramientas.
	2	El sistema mostrará un cuadro de diálogo para navegar por el equipo del usuario.
	3	Seleccionamos la carpeta destino donde vamos a guardar el nuevo objeto de aprendizaje
	4	El objeto de aprendizaje por defecto es un objeto válido mínimo, que se mostrará en la herramienta.
	5	El sistema mostrará un árbol de directorios donde quedará reflejado el objeto de aprendizaje, facilitando de esta forma su edición.
Secuencia alternativa	**Paso**	**Acción**
	5	Si el objeto de aprendizaje por defecto no se localizase, el sistema no mostrará el árbol de representación, sino que alertará de lo ocurrido mediante un mensaje de error.

4.1.3 Caso de uso "Salvar objeto de aprendizaje" (LOME03)

Objetivo en concreto	Guardar la información del objeto de aprendizaje, que se encuentra actualmente abierto en la herramienta, en los ficheros físicos que correspondan.	
Entradas	Modelo Java del objeto de aprendizaje abierto. Fichero físico del objeto de aprendizaje.	
Precondiciones	Debe existir un objeto de aprendizaje abierto en la herramienta LOMEditor.	
Salidas	Fichero imsmanifest.xml que contiene la información del objeto.	
Poscondición si éxito **Poscondición si fallo**	La información del objeto de aprendizaje actualmente abierto se vuelca a fichero físico, quedando guardado. Se muestra al usuario una ventana de error.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede al Menú Principal, Archivo, Guardar, Guardar Objeto de Aprendizaje ó al botón Guardar de la barra de herramientas.
	2	El sistema generará de nuevo el imsmanifest.xml del objeto de aprendizaje, reflejando en él todos los posibles cambios realizados desde la apertura del mismo.
Secuencia alternativa	**Paso**	**Acción**
	2	En caso de no existir un objeto de aprendizaje abierto, o no existir la carpeta destino del objeto, se mostrará un error.

4.1.4 Caso de uso "Comprimir objeto de aprendizaje" (LOME04)

Objetivo en concreto	Guardar la información del objeto de aprendizaje, que se encuentra actualmente abierto en la herramienta, en los ficheros físicos que correspondan, comprimiendo además esta información en formato zip.	
Entradas	Modelo Java del objeto de aprendizaje abierto. Ficheros físicos del objeto de aprendizaje.	
Precondiciones	Debe existir un objeto de aprendizaje abierto en la herramienta LOMEditor.	
Salidas	Fichero .zip que contendrá todo el objeto de aprendizaje, incluidas las últimas modificaciones o recursos añadidos.	
Poscondición si éxito **Poscondición si fallo**	La información del objeto de aprendizaje actualmente abierto se vuelca a fichero físico, quedando guardado y además comprimimos todo el objeto destino. Se muestra al usuario una ventana de error.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede al Menú Principal, Archivo, Guardar, Comprimir Objeto de Aprendizaje ó al botón Comprimir de la barra de herramientas.
	2	Se mostrará un cuadro de diálogo donde se indicará el nombre y destino del nuevo objeto de aprendizaje.
	3	El sistema generará de nuevo el imsmanifest.xml del objeto de aprendizaje, reflejando en él todos los posibles cambios realizados desde la apertura del mismo. Asimismo, generará un nuevo .zip que contendrá el objeto de aprendizaje modificado y todos los ficheros del mismo.
Secuencia alternativa	**Paso**	**Acción**
	2	En caso de no existir un objeto de aprendizaje abierto, o no existir la carpeta destino del objeto, se mostrará un error.

4.1.5 Caso de uso "Abandonar herramienta LOMEditor" (LOME05)

Objetivo en concreto	Cerrar la herramienta, abandonando la edición de objetos de aprendizaje. En caso de encontrarse un objeto abierto actualmente, permite su guardado.	
Entradas	Modelo Java del objeto de aprendizaje abierto. Fichero físico del objeto de aprendizaje.	
Precondiciones		
Salidas		
Poscondición si éxito **Poscondición si fallo**	La herramienta se cierra correctamente. La herramienta no ha conseguido cerrarse correctamente.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede al Menú Principal, Archivo, Salir, ó cierra la ventana principal de la aplicación desde el botón habilitado para ello, en la esquina superior derecha.
	2	La aplicación se cerrará.
Secuencia alternativa	**Paso**	**Acción**
	2	En caso de existir un objeto de aprendizaje abierto, se mostrará un diálogo donde preguntar al usuario si desea guardarlo o no.
	3	Si desea guardar, pasamos al Paso 2 del Caso de Uso LOME04.
	4	Aceptada o no la opción anterior, pasaremos al Paso 2 del Caso de Uso LOME05

4.1.6 Caso de uso "Insertar Nodo en el Árbol del Objeto" (LOME06)

Objetivo en concreto	Insertar un nuevo nodo en el árbol que representa las etiquetas del objeto de aprendizaje que componen el manifiesto.	
Entradas	Modelo Java del objeto de aprendizaje abierto. Nuevo nodo.	
Precondiciones	Debe existir un objeto de aprendizaje abierto.	
Salidas		
Poscondición si éxito	Se inserta un nuevo nodo en el modelo del árbol que representa el manifiesto del objeto de aprendizaje, mostrándose gráficamente en la ventana.	
Poscondición si fallo	Se muestra al usuario una ventana de error, no habiéndose insertado el nuevo nodo.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario sitúa el ratón sobre el nodo del árbol donde desea actuar y presiona el botón derecho.
	2	El sistema mostrará un menú emergente donde se despliegan las diferentes opciones para este nodo, extraídas del modelo generado durante la validación.
	3	Seleccionamos la opción "Nuevo", si es posible, y escogemos el nuevo nodo a insertar.
	4	El sistema generará un nuevo elemento y lo insertará como hijo del nodo seleccionado por el usuario, y por tanto, del elemento que este último contiene y representa.
	5	Se mostrará gráficamente el resultado.
Secuencia alternativa	**Paso**	**Acción**
	3	No existen hijos posibles para el nodo seleccionado, o en este momento no es posible insertar más nodos en el mismo, por lo que no aparecen opciones para nuevo. No se realiza operación alguna.

4.1.7 Caso de uso "Eliminar Nodo del Árbol del Objeto" (LOME07)

Objetivo en concreto	Eliminar un nodo del árbol que representa las etiquetas del objeto de aprendizaje que componen el manifiesto.	
Entradas	Modelo Java del objeto de aprendizaje abierto.	
Precondiciones	Debe existir un objeto de aprendizaje abierto.	
Salidas		
Poscondición si éxito **Poscondición si fallo**	Se elimina un nodo del modelo del árbol que representa el manifiesto del objeto de aprendizaje, mostrándose gráficamente en la ventana. Se muestra al usuario una ventana de error, no habiéndose eliminado el nodo.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario sitúa el ratón sobre el nodo del árbol donde desea actuar y presiona el botón derecho.
	2	El sistema mostrará un menú emergente donde se despliegan las diferentes opciones para este nodo, extraídas del modelo generado durante la validación
	3	Seleccionamos la opción "Eliminar", si es posible.
	4	El sistema eliminará el nodo seleccionado del modelo que representa al objeto de aprendizaje.
	5	Se mostrará gráficamente el resultado.
Secuencia alternativa	**Paso**	**Acción**
	3	El nodo seleccionado no puede ser eliminado, es obligatorio, y por tanto no encontramos esta opción disponible. No se realiza operación alguna.

4.1.8 Caso de uso "Modificar Nodo del Árbol del Objeto" (LOME08)

Objetivo en concreto	Modificar los atributos de un nodo del árbol que representa las etiquetas del objeto de aprendizaje que componen el manifiesto.	
Entradas	Modelo Java del objeto de aprendizaje abierto.	
Precondiciones	Debe existir un objeto de aprendizaje abierto.	
Salidas		
Poscondición si éxito **Poscondición si fallo**	Se modifican los atributos del elemento que representa el nodo seleccionado, en el árbol del objeto de aprendizaje. Se muestra un error y no se realiza la modificación del nodo.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario sitúa el ratón sobre el nodo del árbol donde desea actuar y presiona el botón derecho.
	2	El sistema mostrará un menú emergente donde se despliegan las diferentes opciones para este nodo, extraídas del modelo generado durante la validación
	3	Seleccionamos la opción "Modificar".
	4	El sistema nos mostrará en el Panel de Modificación los diferentes atributos del elemento al que hemos accedido, así como el valor de los mismos, en caso de tenerlo.
	5	Introducimos los valores que deseemos en cada caso.
	6	Pulsamos el botón "Modificar" del panel.
	7	El sistema modificará el elemento que estamos tratando.
Secuencia alternativa	**Paso**	**Acción**
	4	El nodo no contiene atributos modificables, y por tanto no se mostrará nada en el panel.

4.2 Módulo 2: Módulo de composición

4.2.1 Caso de uso "Mostrar objeto de composición" (COMP01)

Objetivo en concreto	Mostrar la parte que nos interesa de un objeto de aprendizaje guardado en formato zip, para realizar la composición entre dos objetos. Esta parte que nos interesa mostrar es el nodo organizations de dicho objeto de aprendizaje, con el que creamos lo que hemos definido como un objeto de composición. (En la composición trabajaremos con este tipo de objetos).	
Entradas	Objeto de aprendizaje.	
Precondiciones	El objeto de aprendizaje a mostrar debe estar comprimido en formato zip, es validado previamente y no ha de estar ya mostrado en el árbol de composición.	
Salidas		
Poscondición si éxito **Poscondición si fallo**	El objeto de composición es mostrado en el árbol de composición, en el panel izquierdo de la aplicación. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de mostrar objeto de composición de la barra de herramientas o en esa misma opción pero en el menú Composición de la barra de menús.
	2	El sistema muestra una ventana en la que el usuario elige el objeto de aprendizaje comprimido que quiere mostrar.
	3	El sistema crea una carpeta temporal con el nombre del objeto dado, en la que descomprime el objeto de aprendizaje y sobre la que trabajará durante la composición del mismo. Si anteriormente se mostró en el árbol de composición un objeto con ese mismo nombre, se muestra un mensaje advirtiendo al usuario que por este motivo (para no sobrescribir la carpeta temporal eliminando posibles cambios realizados y no guardados) el archivo no puede ser abierto.
	4	Si el objeto que se quiere abrir no está mostrado ya en el árbol de composición, el sistema crea un objeto de composición a partir del nodo organizations del objeto de aprendizaje elegido y lo muestra en el árbol de composición.
Secuencia alternativa	**Paso**	**Acción**
	4	Si el objeto que se quiere abrir ya está mostrado en el árbol de composición, se muestra un mensaje de error y no se produce ningún cambio.

4.2.2 Caso de uso "Composición en paralelo" (COMP02)

Objetivo en concreto	Composición en paralelo de dos objetos de aprendizaje, es decir, incluir en una organización del objeto contenedor una organización del objeto contenido.	
Entradas	Dos objetos de composición.	
Precondiciones	Previamente, deben estar mostrados en el árbol de composición al menos dos objetos de composición.	
Salidas	Objeto compuesto en paralelo.	
Poscondición si éxito **Poscondición si fallo**	El objeto ha sido compuesto satisfactoriamente. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de composición en paralelo.
	2	Se muestra un formulario con las organizaciones de cada uno de los objetos de aprendizaje mostrados en el árbol de composición. En este formulario se elige la organización padre (por tanto, el objeto contenedor).
	3	El usuario selecciona una organización padre y pulsa aceptar.
	4	Se muestra un segundo formulario con las organizaciones de los objetos de aprendizaje mostrados en el árbol de composición excepto el objeto elegido como padre en el paso anterior. En este formulario se elegirá la organización hijo (por tanto, el objeto contenido).
	5	El usuario selecciona una organización hijo y pulsa aceptar.
	6	Internamente, el sistema ha realizado la composición de estos objetos, actualizando los datos en la carpeta temporal del objeto seleccionado como padre.
	7	Se actualiza el árbol de composición mostrando el objeto compuesto y eliminando el objeto hijo.
	8	Se elimina también la carpeta temporal que fue creada para el objeto de aprendizaje hijo.
Secuencia alternativa	**Paso**	**Acción**
	2	Si no hay al menos dos objetos de composición mostrados en el árbol de composición, se muestra un mensaje de error, ya que no se puede realizar la composición.
	3.1	Si no se selecciona una organización y se pulsa aceptar, se muestra un mensaje de error.
	3.2	Si se pulsa Cancelar, no se produce ningún cambio y el formulario se cierra.
	5.1	Si no se selecciona una organización y se pulsa aceptar, se muestra un mensaje de error.
	5.2	Si se pulsa Cancelar, no se produce ningún cambio y el formulario se cierra.

4.2.3 Caso de uso "Composición en profundidad" (COMP03)

Objetivo en concreto	Composición en profundidad de dos objetos de aprendizaje, es decir, incluir en un ítem del objeto contenedor una organización del objeto contenido.	
Entradas	Dos objetos de composición.	
Precondiciones	Previamente, deben estar mostrados en el árbol de composición al menos dos objetos de composición.	
Salidas	Objeto compuesto en profundidad.	
Poscondición si éxito **Poscondición si fallo**	Composición en profundidad del objeto realizada correctamente. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de composición en profundidad.
	2	Se muestra un formulario con los items de cada organización de cada uno de los objetos de aprendizaje mostrados en el árbol de composición. En este formulario se elige el ítem padre (por tanto, el objeto contenedor).
	3	El usuario selecciona un ítem padre y pulsa aceptar.
	4	Los pasos siguientes tanto de la secuencia normal como de la alternativa, son los mismos que los explicados en *COMP02*.
Secuencia alternativa	**Paso**	**Acción**
	2	Si no hay al menos dos objetos de aprendizaje abiertos en el árbol de composición, se muestra un mensaje de error, ya que no se puede realizar la composición.
	3.1	Si no se selecciona un ítem y se pulsa aceptar, se muestra un mensaje de error.
	3.2	Si se pulsa Cancelar, no se produce ningún cambio y el formulario se cierra.

4.2.4 Caso de uso "Guardar objeto compuesto como..." (COMP04)

Objetivo en concreto	Guardar un objeto compuesto comprimido en formato zip en el destino seleccionado por el usuario.	
Entradas	Objeto de aprendizaje compuesto.	
Precondiciones	Debe haber al menos un objeto de aprendizaje compuesto.	
Salidas		
Poscondición si éxito	El objeto de aprendizaje compuesto seleccionado por el usuario ha sido comprimido en formato zip en el destino elegido por el usuario.	
Poscondición si fallo	Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de Guardar objeto compuesto como...
	2	Si se ha realizado al menos una composición, se muestra una ventana en la que aparecen todos los objetos que han sido compuestos.
	3	El usuario selecciona el objeto que desea guardar y pulsa aceptar.
	4	Se muestra una ventana de navegación en la que el usuario selecciona el destino en que quiere guardar el objeto compuesto comprimido.
	5	El objeto seleccionado se comprime en la ruta destino dada.
	6	Se muestra un mensaje en el que se informa al usuario que el objeto ha sido guardado comprimido satisfactoriamente en la ruta seleccionada.
	7	Si ha sido compuesto más de un objeto, se pregunta al usuario si desea guardar otro objeto. Si la respuesta es afirmativa se vuelve al paso 2.
Secuencia alternativa	**Paso**	**Acción**
	2	Si no se ha realizado ninguna composición se muestra un mensaje de error.
	3	Si se pulsa Cancelar en la ventana en la que el usuario debe elegir el objeto compuesto que desea guardar, no se realiza ningún cambio.
	4	Si cancela la ventana de navegación de forma que no elige ruta destino, se cierra la ventana y no se produce ningún cambio (el objeto no es guardado).
	7.1	Si no ha sido compuesto más de un objeto, se cierran las ventanas.
	7.2	Si el usuario no desea guardar más objetos, se cierran las ventanas.

4.2.5 Caso de uso "Guardar todo" (COMP05)

Objetivo en concreto	Guardar, comprimidos, en los destinos elegidos por el usuario todos los objetos que han sido compuestos hasta el momento y no han sido guardados.	
Entradas	Objetos de aprendizaje compuestos.	
Precondiciones	Debe haber al menos un objeto de aprendizaje compuesto que no haya sido guardado anteriormente.	
Salidas		
Poscondición si éxito **Poscondición si fallo**	Los objetos de aprendizaje compuestos han sido comprimidos en formato zip. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de Guardar todo (de la composición).
	2	Si se ha realizado al menos una composición, se muestra una ventana de navegación en la que el usuario selecciona el destino en que quiere guardar el objeto compuesto comprimido especificado en la parte superior de dicha ventana.
	3	Si acepta la ventana de navegación, el objeto es comprimido en la ruta destino dada.
	4	Se muestra un mensaje en el que se informa al usuario que el objeto compuesto ha sido comprimido correctamente en el destino dado.
	5	Si hay más objetos compuestos, se vuelve al paso 2.
Secuencia alternativa	**Paso**	**Acción**
	2	Si no se ha realizado ninguna composición o todos los objetos que han sido compuestos ya han sido guardados, se muestra un mensaje de error.
	3	Si cancela la ventana de navegación, el objeto actual no es zipeado, las ventanas se cierran y los objetos que quedaban por guardar no son guardados.
	5	Si no hay más objetos compuestos, se cierran las ventanas.

4.2.6 Caso de uso "Salir de la composición" (COMP06)

Objetivo en concreto	Cerrar la herramienta de composición.	
Entradas		
Precondiciones	Haber mostrado algún objeto de aprendizaje para la composición.	
Salidas		
Poscondición si éxito **Poscondición si fallo**	La aplicación se muestra tal y como estaba al iniciar la aplicación. No se realiza ninguna acción.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón Salir de la composición.
	2	Se informa al usuario de que los cambios realizados se perderán si al salir de la composición no fueron guardados previamente. Se le pregunta si realmente quiere salir y si la respuesta es afirmativa se continúa con la secuencia de pasos normal.
	3	El sistema elimina el árbol de los objetos de composición y la aplicación se muestra tal y como estaba al iniciar la misma.
	4	Se elimina la carpeta temporal que se creó al iniciar las composiciones de objetos.
Secuencia alternativa	**Paso**	**Acción**
	2	Si la respuesta es negativa, (el usuario no quiera salir de la composición) no se realiza ninguna acción.

4.3 Módulo 3: Módulo de evaluación de calidad

4.3.1 Caso de uso "Evaluación manual de objetos de aprendizaje" (BBDD01)

Objetivo en concreto	Evaluación de un objeto de aprendizaje abierto previamente de manera manual, indicando al sistema cada uno de los valores de evaluación para cada etiqueta.	
Entradas	Objeto de aprendizaje	
Precondiciones	El objeto de aprendizaje a evaluar debe haber sido abierto previamente.	
Salidas		
Poscondición si éxito **Poscondición si fallo**	El objeto de aprendizaje y su evaluación han sido guardados en la base de datos de objetos del sistema. Se muestra al usuario una ventana de error.	
Actores	Usuario, la base de datos y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pincha sobre el botón de evaluación manual de objetos de aprendizaje.
	2	Si hay un objeto de aprendizaje abierto el sistema parsea el manifiesto asociado al objeto de aprendizaje.
	3	Si el objeto tiene metadatos se muestra un formulario en el que se solicita al usuario información para la evaluación del objeto de aprendizaje.
	4	El usuario rellena los datos solicitados por el sistema, evaluando cada una de las etiquetas de los metadatos del objeto de aprendizaje y pulsa aceptar.
	5	El sistema muestra por pantalla una ventan resumen, con el resultado de la evaluación general del objeto de aprendizaje.
	6	Si el usuario pulsa aceptar, el objeto de aprendizaje se guarda en la base de datos del sistema al igual que su evaluación.
Secuencia alternativa	**Paso**	**Acción**
	2	Si no hay objeto de aprendizaje abierto se muestra error.
	3	Si el objeto de aprendizaje no tiene metadatos, se muestra una pantalla de error al usuario ya que el sistema de evaluación se basa en los metadatos y no tendría sentido guardar objetos sin metadatos en el sistema.
	6	Si el usuario pulsa cancelar se vuelve al formulario de evaluación manual, sin haberse guardado ni el objeto ni la evaluación en la base de datos del sistema.

4.3.2 Caso de uso "Evaluación automática de objetos de aprendizaje mediante CBR" (BBDD02)

Objetivo en concreto	Evaluación automática de un objeto de aprendizaje previamente abierto por el usuario basándose en la técnica de Inteligencia Artificial conocida como CBR (Case Based Reasoning).	
Entradas	Objeto de aprendizaje.	
Precondiciones	El objeto de aprendizaje a evaluar debe haber sido abierto previamente.	
Salidas		
Poscondición si éxito **Poscondición si fallo**	El objeto de aprendizaje y su evaluación han sido guardados en la base de datos de objetos del sistema. Se muestra al usuario una ventana de error.	
Actores	Usuario, sistema y base de datos.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de evaluación automática basada en CBR.
	2	Si hay un objeto de aprendizaje abierto el sistema parsea el manifiesto asociado al objeto de aprendizaje.
	3	Si el objeto tiene metadatos se muestra una ventana con el resultado de la evaluación CBR al usuario. La evaluación será un número real entre 0 y 4.
	4	Si el usuario pulsa aceptar se guarda el objeto de aprendizaje y su evaluación en la base de datos de la herramienta.
Secuencia alternativa	**Paso**	**Acción**
	2	Si no hay objeto de aprendizaje abierto se muestra al usuario una pantalla de error.
	3	Si el objeto de aprendizaje no tiene metadatos, se muestra una pantalla de error al usuario ya que el sistema de evaluación se basa en los metadatos y no tendría sentido guardar objetos sin metadatos en el sistema.
	4.1	Si el identificador del manifiesto del objeto de aprendizaje ya se encuentra en la base de datos del sistema, se mostrará un mensaje de error al usuario y no se guardará el objeto de aprendizaje, ya que previamente había sido guardado.
	4.2	Si el usuario pulsa cancelar el objeto no se guardará en la base de datos del sistema y se volverá a la herramienta de autoría de objetos de aprendizaje.
	4.3	Si el usuario pulsa cambiar, se accede a la herramienta de evaluación manual de objetos de aprendizaje. Se mostrará, por tanto, el formulario solicitando los datos de evaluación para el objeto de aprendizaje (paso 3, caso BBDD01).

4.3.3 Caso de uso "Consulta a la BBDD de objetos de aprendizaje" (BBDD03)

Objetivo en concreto	Búsqueda de objetos de aprendizaje en la base de datos del sistema.	
Entradas		
Precondiciones		
Salidas		
Poscondición si éxito **Poscondición si fallo**	Lista de objetos de aprendizaje que verifican las restricciones de búsqueda introducidas por el usuario. Se muestra un mensaje de error al usuario.	
Actores	Usuario, sistema y base de datos.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa sobre el botón de consultar base de datos.
	2	El sistema muestra un formulario en el que el usuario debe rellenar diferentes campos para establecer los criterios de búsqueda en la base de datos del sistema. Si el usuario no rellena ningún campo se mostrará todo el contenido de la base de datos. Pulsa aceptar.
	3	El sistema consulta la base de datos y muestra todos los objetos que en ella se encuentren y verifiquen las condiciones de búsqueda impuestas por el usuario.
	4	Si el usuario lo desea puede acceder a la edición de uno de los objetos resultados de la búsqueda con la herramienta de autoría LOMEditor, pinchando en el botón asociado al objeto.
Secuencia alternativa	**Paso**	**Acción**
	3.1	Si no hay objetos de aprendizaje que satisfagan las condiciones de búsqueda solicitadas por el usuario se muestra un mensaje que indica que no hay objetos que verifiquen los criterios de búsqueda.
	3.2	Si se produce algún error en la conexión o consulta con la base de datos del sistema se muestra un mensaje de error al usuario

SECCIÓN 5ª

ARQUITECTURA DEL SISTEMA

5.1 Introducción

A continuación se expone la arquitectura seguida por el proyecto LOMEditor para su implementación. Esta sección presenta detalladamente todos los aspectos de dicha arquitectura, así como las razones por las que se adopta cada diseño.

Para definir la arquitectura de un sistema hay que poner especial atención y cuidado, ya que determinará el comportamiento y futuro del mismo. Aspectos a tener en cuenta para esta definición son los servicios que se desea ofrecer, así como las expectativas de crecimiento o evolución del sistema. No pueden olvidarse, además, los estándares y tendencias más reconocidas actualmente, y que por tanto, marcarán los próximos patrones de diseño. La accesibilidad y reutilización son otros aspectos muy importantes a tener en cuenta.

Como hemos dicho, la utilización de patrones y estándares facilita y mejora la usabilidad del software, haciendo de este una pieza más valiosa. Es además un importante parámetro de medición de calidad del software, y debe por tanto recibir nuestra atención.

En el caso de LOMEditor resulta casi natural enfocar el diseño hacia un conjunto de facilidades o servicios a ofrecer al usuario en el campo de la gestión de objetos de aprendizaje. De esta forma, y tratándose de una aplicación de carácter local, se presenta la posibilidad de crear un conjunto de paquetes internos a este fin, que conformen nuestro nivel de servicios o reglas de negocio.

Dada la importancia que desde el principio se ha dado a la independencia de las gramáticas en que se basan los objetos de aprendizaje, así como el carácter modular de la aplicación, parece adecuado adoptar una arquitectura que favorezca este concepto de separación y reutilización. La flexibilidad y organización en entornos interactivos que ofrece el patrón de diseño Modelo-Vista-Controlador (Model View Controller) hacen de este la elección adecuada para conformar el núcleo de nuestra arquitectura en niveles o capas.

5.2 El patrón de diseño Modelo-Vista-Controlador

Modelo Vista Controlador (MVC) es una arquitectura de software que separa el modelo de datos de una aplicación, la interfaz de usuario, y la lógica de control en tres distintos componentes de forma que las modificaciones en el componente de la vista pueden ser realizadas con un mínimo impacto sobre el componente del modelo de datos. Esto es francamente útil ya que los modelos típicamente tienen alto grado de estabilidad (dependiendo de la estabilidad del dominio del problema que esta siendo modelado), donde es el código de la interfaz de usuario el que normalmente sufre de frecuentes y a veces dramáticos cambios (dependiendo de problemas de usabilidad, la necesidad de soportar clases crecientes de usuarios o simplemente la necesidad de mantener la aplicación presentándose como nueva). No obstante, cualquier trabajo sobre el modelo de datos no debe afectar de forma determinante a la interfaz. Separar la vista del modelo hace que el modelo sea más robusto, pues no son necesarias modificaciones adicionales ante cualquier retoque sobre las vistas.

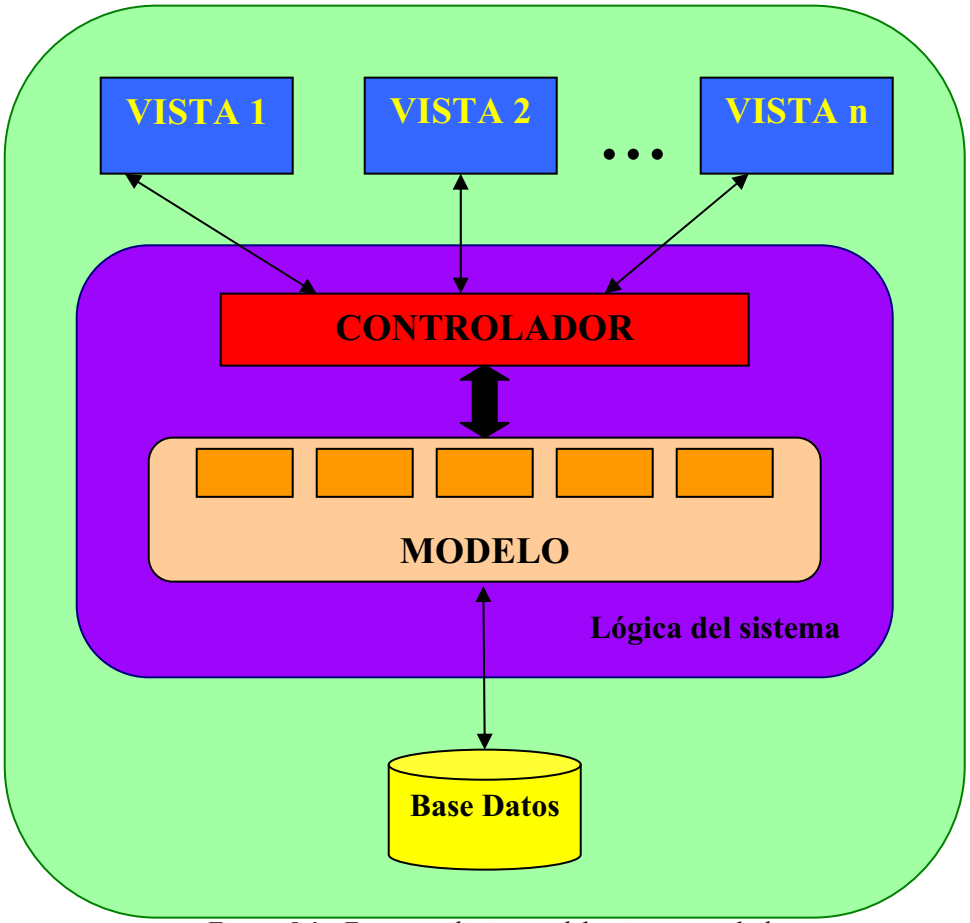


Figura 5.1.- Esquema diseño modelo vista controlador

El patrón fue descrito por primera vez en 1979 por Trygve Reenskaug, quién trabajaba en Smalltalk en los laboratorios de investigación de la Xerox, pero no fue hasta principios de los 90 cuando, de la mano de Java y Sun, paso a ser uno de los más ventajosos diseños en aplicaciones interactivas.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control generalmente es el siguiente:

1. El usuario interactúa con la interfaz de alguna manera (por ejemplo presionando un botón, un enlace, etc.), lo que supone una petición para el sistema.

2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario, además de los datos que este haya podido adjuntar.

3. El controlador accede al modelo, posiblemente actualizando o adaptando los datos enviados por el usuario, lo que aumenta la independencia entre ambos módulos.

4. El modelo realiza la operación requerida por el usuario, y en última instancia por el controlador.

5. El controlador, una vez realizadas las operaciones, delega a los objetos de la vista la tarea de desplegar la interfaz de usuario

6. La vista usa el modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista, en otros casos, como veremos en el siguiente punto, los cambios se reflejan automáticamente.

7. La interfaz espera por nuevas interacciones de usuario para iniciar nuevamente el ciclo.

5.2.1 El modelo delegado

Como se ha explicado anteriormente, la idea principal del patrón de diseño Modelo-Vista-Controlador es lograr la independencia entre la interfaz de usuario (vistas) y el modelo de datos. Sin embargo existen situaciones en las cuales resulta ventajosa la posibilidad de que la vista pueda consultar sin modificar el modelo de datos directamente.

Java ofrece el modelo de legado, mediante los componentes de su paquete gráfico Swing. Estos componentes permiten la creación de interfaz gráfica a partir

de objetos del modelo de datos, siendo este último una constante imagen dentro de la vista; esto significa que cualquier modificación que el modelo realice sobre este elemento se verá automáticamente reflejada en la interfaz.

De esta forma mantenemos el diseño y además optimizamos el trabajo que el sistema realiza para presentar visualmente los resultados.

5.3 Arquitectura por niveles

5.3.1 Descripción general

Esta arquitectura por niveles es la adaptación práctica del patrón de diseño Modelo-Vista-Controlador que hemos adoptado para nuestra herramienta LOMEditor. De esta forma podemos definir y separar claramente todos los elementos que entran en acción en el sistema, lo cual permite una mejor especificación futura, así como una mayor independencia entre los integrantes del equipo de desarrollo.

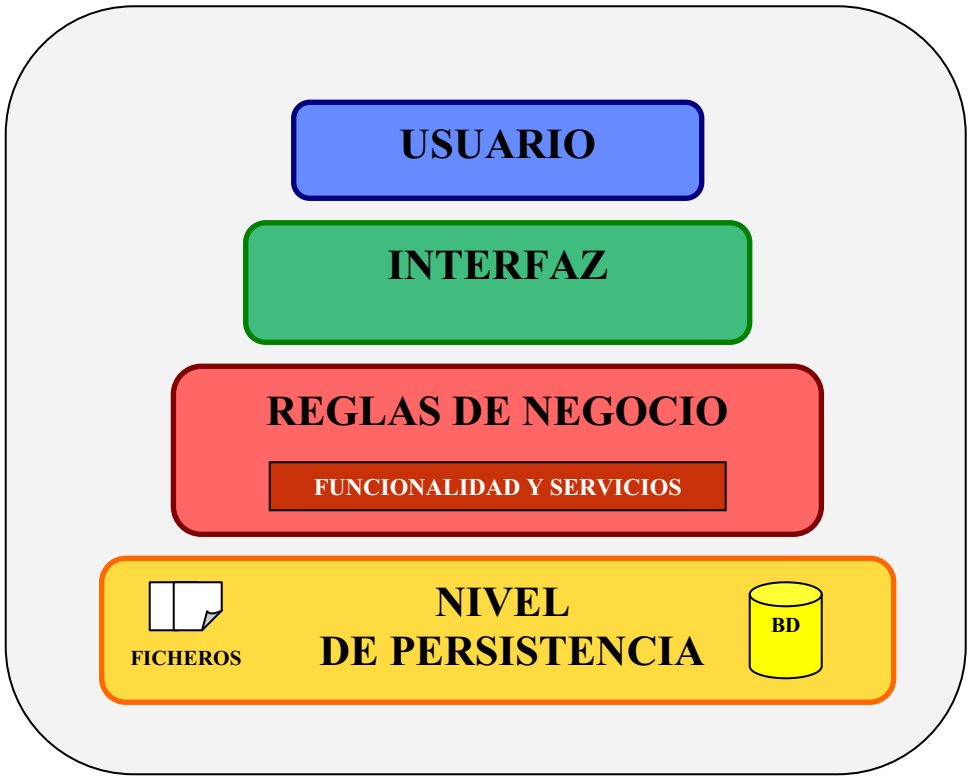


Figura 5.2.- Arquitectura por niveles

Como puede observarse en la figura, existen cuatro niveles o capas diferenciadas, cada una de ellas con una misión muy concreta. Este tipo de arquitecturas presentan una estructura de trabajo muy cómoda, ya que permite a los desarrolladores abstraerse del resto de componentes, una vez definidas las interfaces de interacción entre los mismos.

Los niveles inferiores constituyen las capas más complejas del sistema, que serán las encargadas de prestar los servicios, y mantener la coherencia de los datos sobre los que trabaja el modelo. En estas capas es importante la intervención del sistema operativo residente en la máquina. En nuestro caso, dado que la aplicación es local, vendrá determinado por la especificación.

En los niveles más elevados encontramos las capas que hacen referencia al usuario, y a su interacción con el sistema. Las diferentes vistas que la herramienta pondrá a disposición del cliente tienen su punto de partida en ese nivel de la arquitectura. Son elementos que no requieren una robustez tan elevada como los anteriores, ya que no trabajan directamente con los datos, y en los cuales debe primar la claridad y sencillez de la presentación además de la estética.

Los siguientes puntos tratarán más en detalle las tareas realizadas por cada nivel de la arquitectura.

5.3.2 El usuario y la interfaz

Estos niveles componen las capas más externas del sistema. Por un lado, el usuario se presenta como el principal actor; toda la funcionalidad de la herramienta está destinada a servir las peticiones que este cliente efectúe. Estas peticiones las debe realizar a través de la interfaz de usuario.

El nivel del usuario engloba no solo al teórico cliente de la herramienta, sino todo el entorno que utiliza para acceder a LOMEditor. Dado que se trata de un sistema local, los elementos que intervienen en este primer nivel serán el sistema operativo y el explorador de dicho sistema, así como las aplicaciones requeridas para ejecutar o visualizar los diferentes recursos que presente el objeto de aprendizaje.

El escalón directamente relacionado con el usuario es el nivel de la interfaz. En esta capa se localizan las diferentes vistas que la aplicación ofrecerá al usuario. Este nivel tiene como principal objetivo ofrecer al nivel superior una comunicación con el modelo, de forma que pueda servirse de su funcionalidad mediante unas cómodas vistas que simplifican de cara al usuario cualquier trabajo con los objetos de aprendizaje.

Hay que reiterar que estos niveles superiores deben tener como principal premisa la simplicidad y la estética, dado que serán la única cara que el cliente podrá realmente ver. Es importante para un producto de estas características ofrecer un entorno cómodo, amigable y sencillo donde poder trabajar, sin perder por ello calidad a nivel funcional.

Destacar en este caso las posibilidades que ofrece Java para el trabajo con interfaces de usuario.

5.3.3 Reglas de negocio

En este nivel encontramos los elementos encargados de atender y procesar las peticiones que le hacen llegar las capas superiores. Para ello debe implementar los diferentes servicios necesarios para dar soporte a la funcionalidad ofrecida por la aplicación.

Intervienen aquí el controlador y el modelo de datos. El controlador recibe las peticiones del usuario, y debe en primer lugar tratar la información, en caso de ser necesario, y en segundo lugar debe decidir a que parte del modelo delega esta petición, teniendo en cuenta que elemento implementa el servicio requerido.

Este nivel, por tanto, contiene la lógica del sistema. Será el cerebro de LOMEditor, y debe ponerse especial cuidado en su generación, ya que de ello dependen no solo el comportamiento de la herramienta, sino también las futuras evoluciones de estos servicios. Atendiendo a este fin se ha promovido la utilización de paquetes Java, cada uno de los cuales encargado de un específico campo de trabajo. Estos pequeños subsistemas se explicarán más detalladamente en los detalles de implementación.

Dentro del nivel de Reglas de negocio, por tanto, encontramos un subnivel que engloba la funcionalidad y servicios que la herramienta ofrece. Independizando, como anteriormente se ha dicho, estos paquetes útiles, obtenemos un conjunto de librerías comunes a disposición de cualquiera de los integrantes del equipo de desarrollo.

5.3.4 El nivel de persistencia

Es este el nivel más inferior de la arquitectura. En el se realiza la gestión de los recursos físicos del sistema, ya sean ficheros o registros de la base de datos.

Nuestro sistema, mediante esta capa y bajo instrucciones del modelo de datos, trabaja principalmente con los siguientes ficheros físicos:

4. *Ficheros xml y xsd*

   Estos ficheros, que contendrán la información relevante acerca del objeto de aprendizaje, serán tratados por el paquete de Java JDom. Constituyen el principal recurso físico de LOMEditor, ya que cualquier modificación sobre los objetos de aprendizaje será directamente reflejada en estos ficheros.

5. *Recursos*

Diferentes formatos de ficheros conforman este otro grupo, principalmente compuesto por aquellos archivos que contienen los recursos del objeto de aprendizaje. Su gestión será realizada directamente por la herramienta si es posible, o delegada a las diferentes aplicaciones habitualmente encargadas de ello, en caso de tratarse de tipos no estándar.

6. *Base de datos*

El sistema cuenta con una base de datos relacional que le permite almacenar los diferentes casos útiles para el análisis o evaluación de la calidad. Esta base de datos es soportada por el servidor MySQL. El acceso a esta información se realizará mediante otro paquete de Java, en este caso el que ofrece facilidades de conexión a bases de datos, JDBC.

SECCIÓN 6ª

DETALLES DE IMPLEMENTACIÓN

6.1 Módulo 1: Herramienta de autoría

6.1.1 Introducción

Este punto tratará de describir la implementación de funcionalidad principal de la herramienta LOMEditor, el núcleo de la aplicación. Hay que tener muy presente en todo momento la arquitectura del sistema, explicada en la sección anterior, y seguida fielmente durante el desarrollo de los servicios que a continuación describimos.

El código se encontrará dividido en paquetes, donde encontraremos la funcionalidad necesaria para implementar los servicios a los que la herramienta se compromete. El código principal describe una estructura basada en el patrón de diseño Modelo – Vista – Controlador, como se explica en la Arquitectura del Sistema, el cual divide el código a nivel lógico en esos tres segmentos.

Vemos a continuación con más detalles los aspectos principales del editor LOMEditor 2005.

6.1.2 Validación de objetos de aprendizaje

La validación de objetos de aprendizaje es uno de los puntos fuertes del editor LOMEditor, ya que permite al usuario abrir cualquier tipo de objeto de aprendizaje correctamente construido de acuerdo con una gramática de los estándares.

El proceso de validación comprende los siguientes puntos, que detallaremos a continuación:

6.1.2.1 Apertura y parseo del objeto de aprendizaje

En este punto interviene directamente el paquete JDom, que permite un fácil y completo tratamiento sobre ficheros *xml* y *xsd* (en nuestro caso).

Para abrir el objeto de aprendizaje recurriremos a la utilería de *Zip* implementada con este fin, la cual permite comprimir y descomprimir ficheros en este formato. Una vez descomprimido, tras rellenar los datos referentes a rutas y otros datos relevantes del objeto de aprendizaje, guardado todo ello en la clase *ObjetoAprendizaje*, será el momento de utilizar JDom.

Mediante su capacidad de procesamiento de ficheros de lenguaje de marcado, JDom generará un documento que contendrá, en forma de árbol, la información de la gramática en la que se basa el objeto de aprendizaje.

Ahora debemos recorrer esta gramática e ir rellenando las diferentes estructuras que necesitaremos tanto para validar como para las futuras ediciones sobre este objeto de aprendizaje. Esto lo vemos en el siguiente punto

6.1.2.2 Construcción del modelo de validación

Tras parsear mediante JDom las gramáticas del *Content Package* y de *Metadata*, ahora debemos construir el modelo de validación.

Cada etiqueta de las gramáticas reflejadas en las xsd es procesada y analizada por un determinado grupo de clases; lo vemos a continuación:

- xsd:element: Esta etiqueta será procesada por las clases *TipoElemento* y *TablaTiposElementos*, que guardarán, entre otra información, el nombre y tipo de cada elemento encontrado. Todos estos elementos serán guardados en una tabla para su posterior acceso y utilización.

- xsd:attributeGroup: Para esta otra etiqueta, serán principalmente las clases Atributo y *TablaAtributos* las encargadas del procesamiento. En este caso se trata de analizar y guardar los diferentes atributos encontrados en la gramática, así como los detalles de los mismos.

- xsd:complexType: Las clases Elemento, Nodo y *TablaNodos* serán esta vez las que procesen las etiquetas que guardan la información de los elementos complejos. Será aquí donde encontremos la meta-información necesaria en la edición del objeto de aprendizaje, como los posibles hijos o atributos de cada elemento. Podremos encontrar aquí también los nodos complejos conocidos como Extensiones.

- xsd:simpleType: Esta última etiqueta será procesada por las clases *NodoSimple* y *TablaNodosSimples*, que guardarán, análogamente a lo anterior, la información de los elementos de tipo simple del objeto de aprendizaje.

Para poder utilizar toda esta importante información, recurriremos a la clase *ManifestValidator*, que almacenará dichos datos. Guardará además, por separado, la información de la gramática a utilizar para los metadatos del objeto de aprendizaje.

Análogamente actuaremos para la gramática que detalla la estructura de Metadata, generando para la misma la clase *MetadataValidator*.

6.1.2.3 Algoritmo de validación

El algoritmo de validación comprueba tanto la corrección de los datos de los elementos correspondientes al manifiesto del objeto, como de los que pertenecen a la parte de los metadatos del mismo.

Para realizar esta labor se almacena en tablas la información de la estructura que deberá tener el archivo *imsmanifest.xml*. Esta información se encuentra definida mediante dos gramáticas asociadas a dicho fichero, en forma de archivos *xsd* (*XML Schema Definition*); uno que especifica la estructura que deben seguir los datos del archivo correspondientes al manifiesto, y otro que especifica la estructura de los metadatos. El algoritmo, por tanto, realizará una comprobación por medio de encaje de patrones entre las gramáticas que definen la estructura del manifiesto y el contenido del mismo.

Teniendo en cuenta esto, el algoritmo de validación lee el archivo *xsd* del *Content Package* (gramática que define la estructura del manifiesto), y según lo vamos recorriendo vamos almacenando su contenido (*parsing del manifiesto*).

El algoritmo de validación basa su funcionamiento en dos pilas (clase *Stack*) que irán guardando las etiquetas de los elementos, según se vayan leyendo el archivo *xml* y la información almacenada previamente sobre la *xsd* del *Content Package* del objeto de aprendizaje.

En cada una de estas pilas se irán guardando las etiquetas de los elementos que se vayan leyendo de cada uno de los archivos comentados anteriormente. Si se lee una etiqueta de cierre del elemento se introducirá en la pila un objeto de tipo *String* que contenga “TERMINADO”. De esta forma podemos comparar los objetos que sacamos de ambas pilas sabiendo que son hijos de un mismo padre, es decir, que cuando se lee TERMINADO en alguna de las pilas, significará que el elemento no tendrá más hijos. Por ello, si una pila lee terminado antes que la otra, será porque una de ellas tiene más hijos de ese elemento que la otra pila.

Los pasos que realiza el algoritmo de validación, para comprobar la corrección de un objeto de aprendizaje son los siguientes:

Primeramente, se crean ambas pilas y se insertan en cada una de ellas, el elemento raíz del manifiesto del *xml* y el elemento raíz de la *xsd*, respectivamente. Se crea también una variable *valido* que nos indica si el objeto de aprendizaje es válido hasta el momento, e inicialmente está a *true*.

El algoritmo de validación consiste en un bucle while que no terminará hasta que *valido* sea *false* o alguna de las pilas se vacíe.

Dentro de este bucle se leen los objetos colocados en las cimas de ambas pilas y se comprueba que ambos objetos son del mismo tipo. Para esta comprobación tenemos las siguientes posibilidades:

1. Si ambos objetos son de tipo *String*, significa que hay un "TERMINADO" en la cima de ambas pilas, por lo que el objeto será borrado de ambas y se continuará la validación.

2. Si el objeto obtenido de la *xsd* es de tipo *String* y el objeto obtenido del *xml* es de tipo *Element*, significará que el *xml* no tiene una estructura correcta, ya que tiene más elementos de los que debería contener, según su gramática *xsd*. Por esto, la variable *valido* se pondrá a *false* y la validación terminará. El objeto de aprendizaje no será validado y, en consecuencia, no podrá ser abierto.

3. Si el objeto obtenido de la *xsd* es de tipo *Elemento* y el objeto obtenido del *xml* es de tipo *String*, significará que el *xml* ya ha terminado de leer los hijos y la *xsd* no. En este caso, hay dos posibilidades:

    a. Si el elemento de la *xsd* es obligatorio, el objeto no será validado, puesto que no contendrá un elemento obligatorio.
    b. En caso de que el elemento de la *xsd* no fuera obligatorio, se eliminaría este objeto de la pila y se continuaría con la validación del objeto.

4. Si el objeto de la cima en ambas pilas es un elemento, se comprobará que se trata del mismo elemento. Si se trata de dos elementos diferentes la variable *valido* se pondrá a *false* y la validación terminará. En caso contrario, se pasará a la validación de los atributos de dicho elemento. Para ello, se tendrán en cuenta los siguientes aspectos, que serán comprobados para cada atributo:

    a. Los atributos que aparecen como requeridos en la *xsd*, tienen que estar obligatoriamente en el elemento correspondiente en el *xml*.
    b. Cada atributo debe tener el tipo que le corresponde.
    c. El atributo ID no debe estar repetido para los elementos del documento.
    d. Comprobar que el atributo IDREF realmente hace referencia a un ID existente en el documento.

    Después de haber validado los atributos del elemento del *xml*, quitamos el elemento evaluado de la cima de la pila del *xml*. Si el elemento de la *xsd* sólo puede tener una ocurrencia se elimina también de la pila *xsd*.

El siguiente paso es el de evaluar los hijos de los elementos anteriores. Para ello, si la *xsd* indica que el elemento puede tener más hijos se introduce el código de finalización "TERMINADO" en ambas pilas, y a continuación se añaden los hijos correspondientes a cada una de las pilas y se continúa con la evaluación.

Los pasos anteriores indican la secuencia de validación del manifiesto sin tener en cuenta los metadatos del mismo. Debido a que los metadatos del objeto se encuentran dispersos como hijos de diferentes elementos en el manifiesto, y su estructura queda definida mediante otra gramática diferente, al llegar a un elemento *metadata*, si existiera, éste sería validado de manera independiente, ya que esta validación se realizaría mediante otro archivo *xsd* (gramática de metadatos). No obstante, la forma de realizar la validación de la estructura de los metadatos del objeto es idéntica a la comentada anteriormente.

Para facilitar la comprensión del algoritmo de validación a continuación se muestra un ejemplo, indicando el estado de las pilas de validación en cada momento. Para no hacer este desarrollo demasiado complejo, tan sólo se muestra la validación de un objeto sencillo que no contiene metadatos. Por tanto, la validación consistirá en el encaje de patrones entre el fichero *imsmanifest.xml* y su gramática asociada, en este caso, *imscp_v1p1.xsd,* que se muestran a continuación.

imsmanifest.xml

```
<?xml version="1.0"?>
<manifest identifier="MANIFEST1" xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_v1p2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1p3.xsd
http://www.imsglobal.org/xsd/imsmd_v1p2 imsmd_v1p2p2.xsd ">
  <organizations>
      <organization identifier="TOC1">
          <title>xR 500 Robotic Arm</title>
          <item identifier="ITEM1" identifierref="RESOURCE1">
              <title>Overview</title>
          </item>
      </organization>
  </organizations>
  <resources>
      <resource identifier="RESOURCE1" type="webcontent" href="robot2/background/robotintro.htm">
          <file href="robot2/background/robotintro.htm"/>
      </resource>
  </resources>
</manifest>
```

imscp_v1p1.xsd

Este fichero almacena la estructura del manifiesto según la especificación *IMS Content Package v1.1.4*. Previa a la ejecución del algoritmo de validación, la estructura del manifiesto definida en este archivo habrá sido almacenada en las

clases implementadas a estos efectos, tal y como se indica en el punto *6.1.2.2 Construcción del modelo de validación*, en esta misma sección.

Funcionamiento del algoritmo:

Inicialmente se cargarán las raíces correspondientes a ambos documentos, las pilas contendrán, por tanto:

Figura 6.1.2.1.- Paso 1

Al contener la cima dos Elementos se comprueba que sean iguales y se validan sus atributos. Si la validación de los atributos es correcta, se elimina el elemento de la cima de la pila XML y el de la pila XSD, sólo si se trata de un elemento que puede aparecer una única vez en el documento. Como manifest sólo puede aparecer una vez, según esta especificación, se eliminará el elemento cima de ambas pilas y se introducirá "TERMINADO". A continuación, se introducirán en la pila XML los hijos del elemento manifest y en la pila XSD los posibles hijos del mismo, según la gramática que se encuentra almacenada en el modelo de validación.

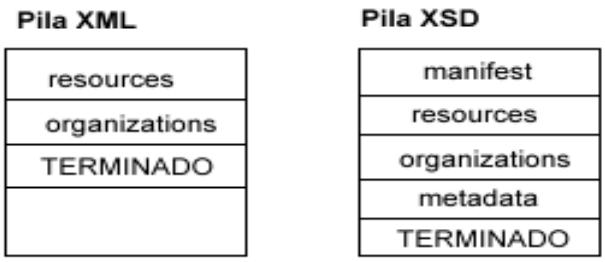


Figura 6.1.2.2.- Paso 2

El siguiente paso en la validación se encontraría con dos elementos en las cimas de ambas pilas, pero de naturaleza diferente. El algoritmo, comprobará entonces que el elemento de la cima de la pila XSD no es obligatorio. Como los submanifiestos no son obligatorios según la gramática utilizada se eliminará de la cima de la pila XSD el elemento manifest.

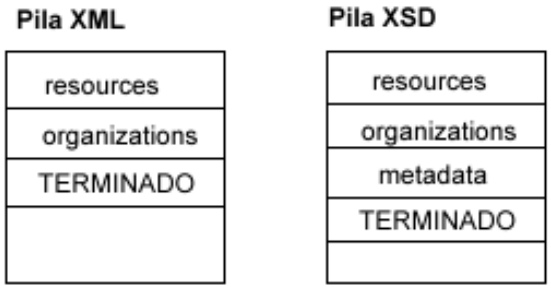


Figura 6.1.2.3.- Paso 3

Llegados a este punto, se procedería a la validación de los elementos resources que se encuentran en la cima de ambas pilas. De nuevo, al tratarse de elementos complejos, con hijos, y del mismo tipo, se procedería a comprobar que los atributos del elemento resources del manifiesto son correctos.

Como en este caso los atributos del elemento resources son correctos, se continuaría evaluando los hijos de este elemento. Con esta finalidad, se introduciría el código "TERMINADO" y los elementos de hijos de resources según la gramática y según el manifiesto. Como el elemento resources sólo puede aparecer una vez dentro del manifiesto se eliminará de la cima de ambas pilas, quedando éstas como indica la siguiente figura.

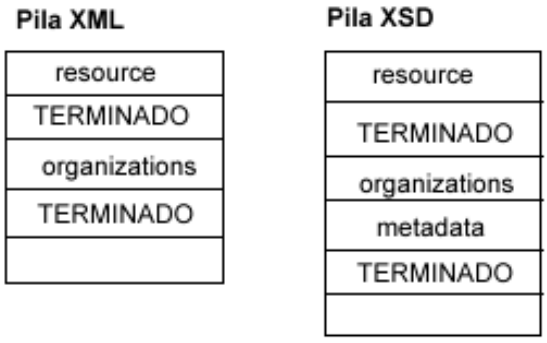


Figura 6.1.2.4.- Paso 4

Al encontrarse con la situación de la figura anterior, el objeto procedería de nuevo a la evaluación de los hijos de los elementos de la cima de la pila, resource, eliminando el elemento en la cima de la pila XML, pero no haciéndolo en la pila XSD ya que este elemento puede aparecer varias veces, según la gramática que se está considerando.

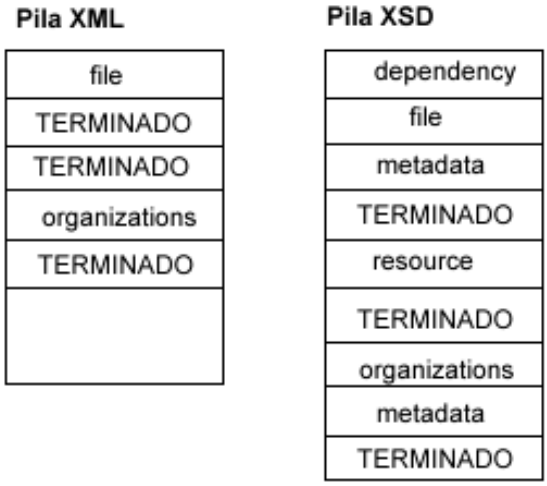


Figura 6.1.2.5.- Paso 5

En este punto el algoritmo, detectaría que dependency no es un elemento obligatorio según la estructura definida mediante la *xsd*, por lo que sacaría el elemento de la cima de la pila XSD y continuaría evaluando. Después comprobaría la correspondencia entre los dos elementos file que se encontraría en la cima de la pila. Puesto que el elemento file del *xml* de ejemplo es correcto, después de una serie de pasos nos encontraríamos en la siguiente situación:

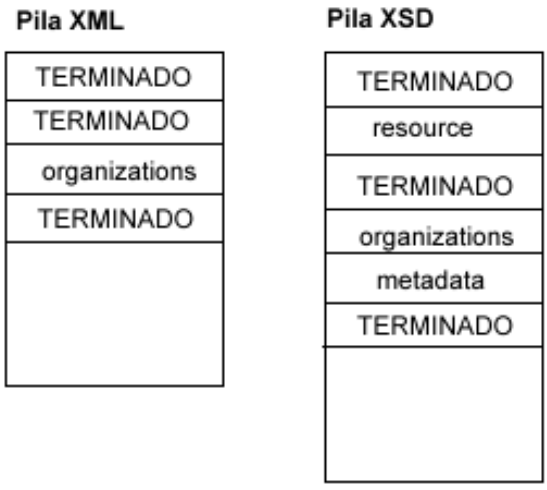


Figura 6.1.2.6.- Paso 6

Los “TERMINADO” que se encuentran en la cima de ambas pilas serían desapilados en este punto.

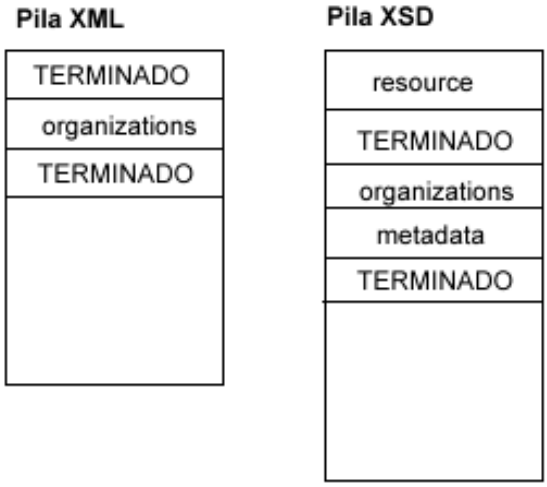


Figura 6.1.2.7.- Paso 7

Al encontrarse en la situación que indica la figura anterior el algoritmo eliminaría el elemento resource de la cima de la pila XSD ya que, aunque es un elemento obligatorio, ya ha aparecido con anterioridad en el fichero *xml*.

El proceso continuaría hasta que el algoritmo encontrase las dos pilas vacías, ya que en el caso de ejemplo el manifiesto valida la *xsd* que define su estructura.

6.1.3 Construcción del modelo de datos

Una vez finalizado el proceso de validación explicado anteriormente, la herramienta se encuentra en disposición de construir el modelo de datos con el que va a trabajar a partir de ahora.

Es, por tanto, una premisa obligatoria a cumplir la correcta validación del objeto, donde se habrá generado el modelo de validación. Partiendo de aquí, vamos en primer lugar a repasar la información que nuestro modelo de datos debe recoger para el correcto funcionamiento del editor.

6.1.3.1 Contenido del modelo de datos

El contenido que nuestro modelo de datos debe completar a partir del objeto de aprendizaje introducido es el siguiente:

7. *Modelo de validación*

Como se ha explicado con anterioridad, ninguna operación estará previamente predefinida en el LOMEditor, es necesario por tanto guardar esta estructura, que contendrá la información de la gramática que describe el objeto de aprendizaje que estamos tratando. En todo momento deberemos recurrir a él para extraer los datos a utilizar en la edición del objeto.

No nos extenderemos más sobre este punto, explicado en el apartado anterior con más detalle. El modelo de datos contendrá una referencia al mismo.

8. *Documento del manifiesto*

Representará el archivo *imsmanifiest.xml* del objeto de aprendizaje. Parseado por JDom, recuperamos un objeto de tipo Document (esto se hace ya para la validación), que será nuestra referencia al modelo real, de donde se tomarán los datos del objeto, y donde se guardarán las modificaciones. Su estructura consiste en un conjunto de objetos *Element* distribuidos en forma del árbol, que representan por tanto las etiquetas del XML y su distribución el dicho documento.

9. *Árbol del objeto de aprendizaje*

Se trata de un árbol temporal que representará al objeto de aprendizaje. Este es el objeto principal del modelo de la aplicación. Será un Árbol Java (*DefaultTreeModel*). Su construcción supone la unión de los datos reales del Document, con los metadatos obtenidos del modelo de validación; de esta forma, para cada Element del Document generaremos un *NodoXML*, que enriqueceremos con la información que el modelo de validación nos da del mismo.

El resto de elementos que intervienen en el modelo de datos lo hacen de forma auxiliar, o para realizar soporte a los servicios descritos en los siguientes apartados, donde serán debidamente explicados.

En segundo lugar veremos el proceso que hemos seguido para rellenar este modelo de datos.

6.1.3.2 Contenido del modelo de datos

Veremos paso a paso el proceso a completar para conseguir generar el modelo de datos correctamente.

1. *Se construye un objeto del tipo Arbol (importante: ver clase Arbol)*

    a. Guardamos el Document generado para la validación.

    b. Guardamos el Modelo de Validación

    c. Generamos recursivamente el Árbol Java

2. *Generación del Árbol Java*

    a. De manera recursiva vamos recorriendo el Document en profundidad, donde para cada Element iremos creando un NodoXML. Para la construcción de este NodoXML necesitaremos la información del Element que representará, y la meta-información de dicho Element que nos ofrece el Modelo de Validación.

    b. Enlazamos los NodoXML de forma que su estructura imita la del árbol del Document.

3. *Construcción del NodoXML (Importante: ver clase NodoXML)*

    a. NodoXML será una clase que extenderá de *DefaultMutableTreeNode*, de forma que permite construir un DefaultTreeModel directamente (el árbol de la clase *Arbol*) y aprovechar todos los métodos de trabajo sobre nodos y árboles (inserción, eliminación, recorrido, etc.)

    b. Buscaremos la información del Elemento al que representa. Para ello nos dirigiremos al modelo de validación. Dicho modelo, referenciado en nuestro modelo de datos, permite la búsqueda y recuperación de Elementos a partir de su nombre, aunque debemos indicar, eso si, sobre que tabla realizar la búsqueda. Nodos Simples o Complejos del manifiesto o de metadata serán nuestros objetivos. Para saber donde buscar cada nodo padre contiene un flag que indica a que grupo pertenece.

    c. Guardaremos la información relevante del nodo.
        i. Nombre del elemento

ii. Referencia al Element, para poder modificar el Document en tiempo real
iii. Lista de posibles hijos del nodo: Cada uno indicando si es obligatorio y sus máximas apariciones.
iv. Lista de posibles Atributos del Nodo: Indicando si son obligatorios y sus valores por defecto.
v. Flag que nos indica si el nodo es Simple, o Complejo, donde además puede resultar ser una Extensión.

4. *Organización del árbol (DefaultTreeModel) de la clase Arbol*

   a. En la generación recursiva del árbol, se van insertando los NodoXML en el mismo orden en que aparecen como etiquetas en el *imsmanifest* (en el Document), para poder ofrecer una correcta vista del mismo al usuario.
   b. Esta organización permite además que cada NodoXML contenga una información muy importante: Sus hijos actuales. Como es un árbol Java y los NodoXML son DefaultMutableTreeNode enriquecidos, contamos con todos los datos que nos puede ofrecer una estructura en árbol, y por tanto, no necesitamos que NodoXML gestione una lista de hijos actuales. Esto nos hace ganar eficiencia, pues se trata de una misma solución para dos problemas (visualización y edición).

5. *Construcción final del JTree:*

   a. La última ventaja de este diseño es la facilidad de trabajo Modelo-Vista que ofrece el modelo delegado de Swing. Esto es, sus objetos gráficos, en este caso JTree, pueden ser construidos a partir de un modelo interno (nuestro *DefaultTreeNode*), de forma que podrá ser visualizada cualquier modificación sobre el mismo, sin tener que realizar operación alguna sobre el JTree. Además, JTree puede proporcionar métodos interesantes para la captura de eventos que permitirá el trabajo sobre el modelo interno. Para más detalles sobre ello es conveniente acudir a la sección de Arquitectura del Sistema.

Como ejemplo del trabajo entre el modelo de datos y el modelo de validación vamos a ver un extracto de código donde se realiza la configuración de un nodo.

```
/**Método que configura los atributos del nodo es decir, las
características de la etiqueta
* a la que representa*/
private void configurarNodo(ManifestValidator modelo)
{
//Caputar las propiedades de la etiqueta que
// representa en el modelo de validación
```

```
MetadataValidator metadataVal;
TablaNodos tablaNodos;
Nodo nodoValidacion=null;
TablaNodosSimples tablaNodosSimple;
NodoSimple nodoValidacionSimple=null;
boolean esHijoExtension=false;

boolean isTipoManifest =
!elemento.getNamespacePrefix().equalsIgnoreCase("imsmd");
if(elemento.getName().equalsIgnoreCase("lom"))
       isTipoManifest=false;

metadataVal = modelo.getMetadataValidator();
//Extraemos del modelo la tabla de nodos complejos
if(isTipoManifest)
{
       tablaNodos = modelo.getNodeTable();
       nodoValidacion = tablaNodos.getNode(nombre);
}
else
{
       // Es un nodo complejo del manifiesto
       // y pertenece a metadata
       tablaNodos = metadataVal.getNodeTable();
       nodoValidacion = tablaNodos.getNode(nombre);
}
if (nodoValidacion != null)
{
       //El nodo es complejo, extraemos sus datos
       tipoNodo = 1;
       this.setAllowsChildren(true);
       //Posibles hijos
       ArrayList posiblesHijos = nodoValidacion.getElementosHijos();
       for (int i = 0; i<posiblesHijos.size(); i++)
       {
              Object oAux = posiblesHijos.get(i);
              //Hijo Elemento
              if(oAux instanceof Elemento)
              {
                     esHijoExtension=false;
                     Elemento eAux = (Elemento) oAux;
                     //Valores por defecto
                     String maxOc = "1";
                     String minOc = "1";

                     if (eAux.getMinOccurs() != null)
                     {
                            minOc = eAux.getMinOccurs();
                     }
                     if (eAux.getMaxOccurs() != null)
                     {
                            maxOc = eAux.getMaxOccurs();
                     }
              Elemento e = new
              Elemento(eAux.getName(),eAux.isTipoManifest(), minOc, maxOc);
              elementList.add(e);
              }
       //Hijo Extension
       else
       {
              esHijoExtension=true;
```

```
                Extension eAux = (Extension) oAux;
                //Posibles atributos
                TablaAtributos posiblesAtributos = eAux.getTablaAtributos();
                Enumeration elems = posiblesAtributos.getElements();
                while (elems.hasMoreElements())
                {
                        Atributo aAux = (Atributo) elems.nextElement();
                        String req = "";
                        if (aAux.isRequired())
                        {
                                req = "required";
                        }
                        Atributo a = new Atributo(aAux.getAttributeGroup(),
                        aAux.getNameAttribute(),
                        aAux.getType(), aAux.getDefault(),
                        req, null);
                        attributeList.add(a);
                        // Si es obligatorio, debemos agregarlo
                        if (aAux.isRequired() && atributoNoCreado(aAux))
                        {
                        String defecto = aAux.getDefault();
                        if (defecto == null)  defecto = "";
                        elemento.setAttribute(aAux.getNameAttribute(),
                defecto);
                        }
        }
        tipoNodo = 2;
        base = eAux.getBase();
        }
}
if(nombre.equalsIgnoreCase("metadata"))
{
        Elemento e = new Elemento("lom",false, "0", null);
        elementList.add(e);
}
//Posibles atributos, si no es un hijo de tipo Extension
if(!esHijoExtension)
{
        //Posibles atributos
        ArrayList posiblesAtributos = nodoValidacion.getAttributes();
        for (int j = 0; j < posiblesAtributos.size(); j++) {
                Atributo aAux = (Atributo) posiblesAtributos.get(j);
                        String req = "";
                if (aAux.isRequired()) {
                req = "required";
        }
        // Puede contener un nodo simple
        NodoSimple nSimpAux = aAux.getNodoSimple();
        NodoSimple nSimp = null;
        Atributo a = new Atributo(aAux.getAttributeGroup(),
        aAux.getNameAttribute(), aAux.getType(),
        aAux.getDefault(), req, null);
        attributeList.add(a);
        // Si es obligatorio, debemos agregarlo
        if (aAux.isRequired() && atributoNoCreado(aAux))
        {
                String defecto = aAux.getDefault();
                if (defecto == null)
                defecto = "";
                elemento.setAttribute(aAux.getNameAttribute(), defecto);
        }
        }
```

```
}
else
{
	if(!isTipoManifest)
	{
		// El nodo es Simple
		tablaNodosSimple = metadataVal.getSimpleNodeTable();
		nodoValidacionSimple = tablaNodosSimple.getSimpleNode(nombre);
	}
	else
	{
		// Se trata de un nodo simple, pero no de Metadata
		tablaNodosSimple = modelo.getSimpleNodeTable();
		nodoValidacionSimple = tablaNodosSimple.getSimpleNode(nombre);
	}
	if (nodoValidacionSimple != null)
	{
		tipoNodo = 0;
		base = nodoValidacionSimple.getBase();
	}
}
}
```

Finalmente habremos conseguido construir un modelo de datos robusto y completo, que permita al usuario realizar las tareas de edición adaptándose totalmente a la gramática que describe el objeto.

6.1.4 Visualización de contenido de objetos de aprendizaje

Este es uno de los aspectos tecnológicos más importante de la herramienta al margen de los relacionados con el trabajo sobre las especificaciones y estructura de objetos de aprendizaje.

De esta forma, la visualización de contenidos que realiza LOMEditor 2005 es independiente de la herramienta, ya que basa su funcionamiento en la generación de una página Web HTML aplicando una XSL al manifiesto del objeto de aprendizaje, diseñada a tales efectos. Con el uso de esta tecnología se consigue mantener la visualización de contenidos independiente de la herramienta. Además, el formato de esta Web generada se aplicará mediante una CSS, lo que mantendrá una independencia también entre contenidos de la visualización, página HTML generada, y estilo de la misma, CSS (*Cascading Style Sheet*).

No obstante, existe un gran inconveniente que restringe la potencia de estas técnicas, que es la no existencia de una API de Java de software libre que permita la visualización correcta y completa de los ficheros que se pueden generar mediante las tecnologías anteriormente comentadas. El recurso por el que se optó finalmente fue la utilización de un *JEditorPane*, el cual permite realizar acciones básicas sobre un HTML, pero no es capaz de recoger otras como, por ejemplo, contenidos dinámicos de la página, Javascript, algunas de las etiquetas para CSS, etc.

La información que se mostrará en este panel será la relacionada únicamente con el contenido del objeto de aprendizaje, es decir, la información de los recursos que contiene el mismo.

El siguiente esquema muestra el efecto deseado con este tipo de implementación:

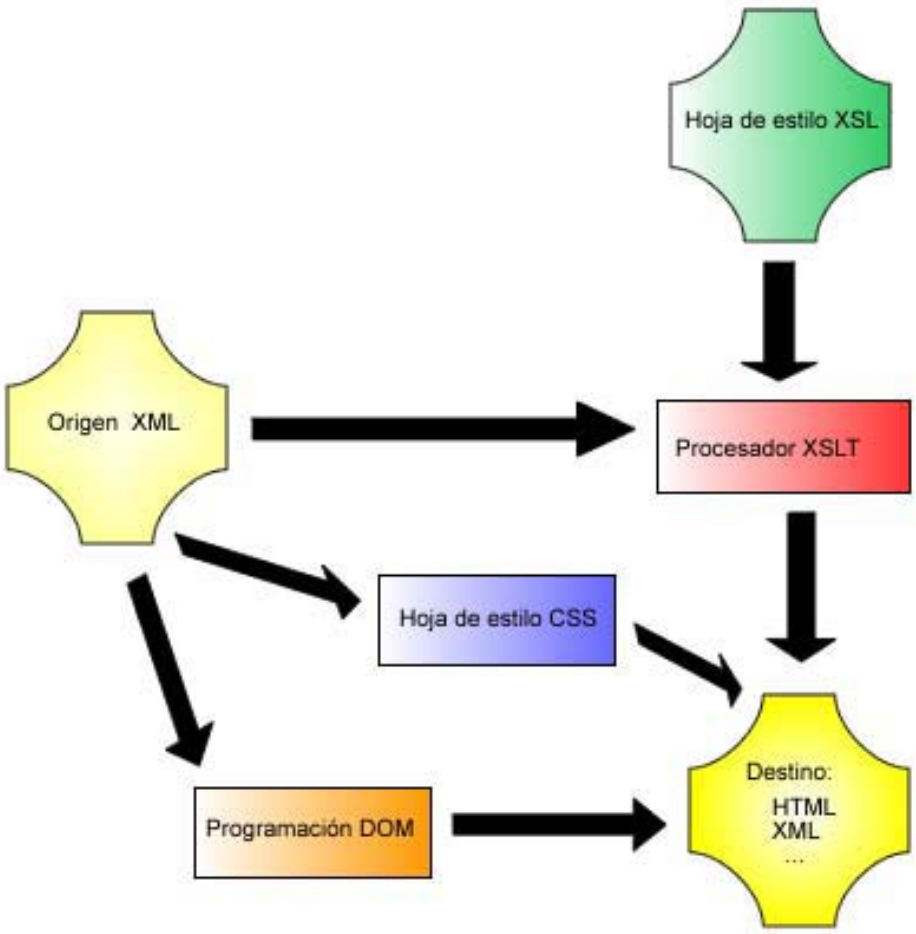


Figura 6.1.4.1.- Transformación de documentos XML

6.1.4.1 Panel de visualización de contenidos

El panel de visualización de contenidos del objeto de aprendizaje, está dividido en dos marcos: marco izquierdo, en el que se muestra un índice de contenidos del objeto o índice de recursos, y el marco central en el que se expone el contenido de cada uno de los recursos del objeto que se indexan en el índice

anterior. Para llevar a cabo esta división se generaron una serie de ficheros *html* que se comentarán a continuación:

1. Fichero viewFrames.html

Este fichero es el que se visualizará en el JEditorPane tras la gestión de visualización de contenidos del objeto de aprendizaje. No obstante, este fichero no se genera dinámicamente, sino que es un fichero estático que se encarga de contener a otros dos, los que formarán el marco izquierdo y el central del panel de visualización de contenidos.

(*Véase Apéndice C.1.3 Fichero viewFrames.html).*

2. Fichero lome.html

Se trata de otro fichero estático que se encarga de mostrar la imagen de la herramienta LOMEditor 2005. Simplemente, aparece al abrirse un objeto de aprendizaje en el marco central de visualización de contenidos, ya que aún no se habrá pinchado sobre ningún recurso a visualizar.

(*Véase Apéndice C.1.4 Fichero lome.html).*

3. Fichero view.html

Éste es el fichero más importante de todos en cuanto a lo que se refiere a visualización de contenidos del objeto de aprendizaje. Se trata de una vista en forma de índice de los recursos del objeto, que será generada de manera dinámica, para cada objeto, a partir de su manifiesto.

(*Véase Apéndice C.1.5 Fichero view.html).*

4. Fichero objetoAprendizaje.css

Este archivo contiene información sobre el aspecto de las páginas HTML. Para evitar que cada página tenga un estilo definido en su fichero HTML, se asociará este archivo a todas las páginas anteriormente comentadas, obteniendo un aspecto global similar en cada una de ellas y manteniendo en los ficheros HTML tan sólo información sobre el contenido de la página y no sobre su estilo.

(Véase Apéndice C.1.2 Fichero objetoAprendizaje.css).

6.1.4.2 Transformación XSL

Como ya se ha comentado anteriormente, el proceso de visualización de un objeto de aprendizaje se basa en la aplicación de una XSL (*Extensible Stylesheet Language*) al manifiesto del objeto de aprendizaje, *imsmanifest.xml*, en el que se recoge la estructura del mismo. Como resultado de esta transformación XSL se obtendrá un fichero HTML, *view.html*, diferente para cada objeto de aprendizaje dependiendo de su manifiesto.

La generación de este HTML de índice de contenidos se realizará atendiendo a las siguientes indicaciones:

1. La disposición del índice de contenidos del objeto de aprendizaje será la especificada en el propio manifiesto del objeto de aprendizaje, según indican las organizaciones de que consta éste. De esta forma, se mostrarán los ítems contenidos en cada una de las organizaciones, y un enlace asociado a ellos que nos lleve al recurso al que apunta cada ítem.

2. Cada una de las organizaciones quedará resaltada, para diferenciarla de las demás, y englobará todos sus recursos asociados. Estos recursos aparecerán con una sangría mayor que la de su organización o su ítem padre, simulando una estructura de directorios.

Para conseguir estos objetivos la plantilla XSL actúa de forma recursiva sobre el árbol que representa el XML del manifiesto del objeto de la siguiente manera:

1. Por cada organización encontrada en el objeto de aprendizaje genera una tabla, con título el nombre de la organización y contenido el resultado de aplicar a la lista de sus ítems hijos.

2. Para cada uno de los ítems, la plantilla XSL generará una lista, en la que cada uno de sus elementos serán enlaces a los recursos asociados a cada ítem. Para lograr este efecto, es necesario realizar un salto en el objeto a la parte del manifiesto en la que se especifican sus recursos y comprobar que el atributo *identifier* del recurso y el *identifierref* del ítem coinciden. Si esto ocurre, el recurso que estamos analizando está apuntado por el ítem del que se proviene y por tanto se generará un enlace html, con destino el contenido del atributo *href* del fichero asociado a dicho recurso.

De esta manera, por cada ítem se tendrá: título del ítem y enlace al recurso al que apunta el ítem.

Este paso se aplicará recursivamente a cada uno de los ítems que contenga el objeto de aprendizaje. De este modo, si un ítem tiene a su vez más ítems contenidos en su interior éstos y sus recursos asociados serán también mostrados.

3. Por último, la plantilla será aplicada recursivamente, al igual que en caso anterior, a todas las etiquetas *manifest* que contenga el manifiesto del objeto. Así, si se trata de un objeto de aprendizaje con manifest anidados o sub-manifest también se mostrará el contenido de éstos.

Para una mejor comprensión de la actuación de la plantilla sobre el fichero *imsmanifest.xml*, véase el *Apéndice C.1.1 Fichero objetoAprendizaje.xsl*

6.1.4.3 Implementación de la transformación XSL desde Java

La clase encargada de llevar a cabo la tarea de realizar la transformación XSL es la clase llamada *ProcesadorXSLT (Véase Diagrama A.1.1.1 del Apéndice A).*

Esta clase tan solo se encarga de, dados el XML del manifiesto y la XSL que se aplicará al mismo, generar el fichero *view.html* aplicando la XSL al XML.

6.1.4.4 Panel de visualización de contenidos: JEditorPane

Como ya se comentó previamente en este documento, el contenido del objeto de aprendizaje, una vez aplicada la XSL correspondiente y generados los ficheros necesarios, será mostrado en forma de fichero HTML que se cargará en un panel de tipo *JEditorPane.*

Para que la información contenida en el panel cambie como cambiaría en un navegador Web según el usuario pulsa a los enlaces de que ésta consta es necesario implementar un Oyente asociado al panel de tipo *HyperlinkListener.* Este oyente será el encargado de responder a las pulsaciones sobre los enlaces de las páginas mostradas en el *JEditorPane*, que serán los enlaces correspondientes a los recursos y que se mostrarán en el marco central del panel.

Debido a las limitaciones en la carga de distintos tipos de ficheros, este oyente realiza además la labor de restringir los formatos de ficheros que se mostrarán en el *JEditorPane*, de modo que, si uno de los recursos sobre los que se pincha no es soportado por el panel se lanza el navegador por defecto del sistema operativo del usuario, y se muestra el recurso en él, si el usuario lo desea.

6.1.4.5 Resumen

La visualización de contenidos de la herramienta es uno de los puntos fuertes del sistema, ya que se mantiene de manera independiente al sistema. La utilización de lenguajes de marcado para el procesamiento del manifiesto del objeto de aprendizaje permite que se pueda modificar este sub-módulo de la herramienta sin realizar cambios en el código Java de la misma, o aumentar el número de tipos de visualizaciones posibles a partir de un mismo objeto de aprendizaje. Así, se podrían realizar diferentes plantillas XSL para cada tipo de visualización deseada. Por ejemplo, se podrían crear distintas plantillas de procesamiento XSL para cada uno de los perfiles de usuarios de los que la aplicación cuenta. Así para cada perfil, se mostraría el contenido del objeto de aprendizaje adaptándolo al tipo de trabajo que realiza este usuario con la herramienta.

Otro aspecto importante, es la utilización de una plantilla CSS, que será la encargada de unificar el aspecto de cada una de las páginas que se generan para la visualización de contenidos del objeto de aprendizaje. Este, por tanto, es otro punto que se podría modificar de manera sencilla, sin tener que modificar el código de la aplicación. Si quisiésemos, por ejemplo, realizar varias plantillas de estilo, o modificar la anterior plantilla a causa de un cambio en el marketing de la herramienta, tan sólo será necesario modificar la plantilla *objetoAprendizaje.css.*

Para facilitar la comprensión de este complejo trabajo de visualización el siguiente esquema trata de mostrar cada uno de los pasos y elementos del sistema encargados de realizarlo:

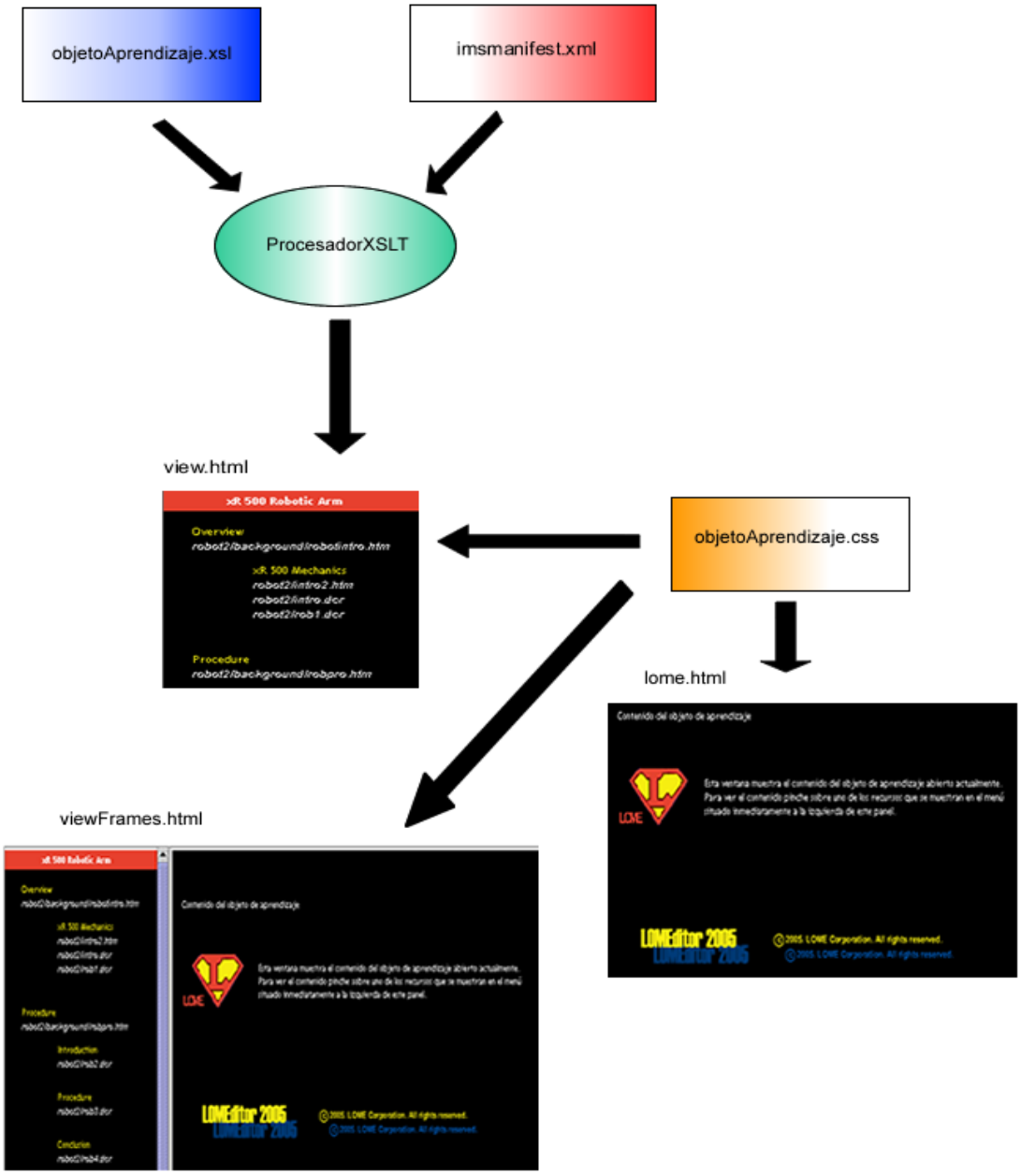


Figura 6.1.4.2. – Proceso de visualización de contenido

6.1.5 Métodos de edición

La herramienta dispone de diversos servicios que ofrece a sus usuarios de cara a la edición de objetos de aprendizaje. Cada uno de ellos conlleva un proceso de trabajo interno importante, que a continuación trataremos de describir.

1. Inserción de un nuevo elemento en el objeto de aprendizaje

Para insertar un nuevo elemento en la estructura de árbol del objeto de aprendizaje debemos en primer lugar analizar el elemento en el cual deseamos insertar el nuevo hijo.

Al analizar este elemento padre podremos extraer la lista de posibles hijos disponibles para la inserción. Cabe la posibilidad de que no encontremos ninguno, ya que las especificaciones describen elementos que no contienen descendencia, o simplemente por temas de limitación de apariciones de los mismos. La lista que contiene estos datos figura como atributo de la clase NodoXML que representa al elemento.

Una vez hayamos escogido el hijo que deseamos insertar, necesitaremos la información que rodea al mismo. Para ello nos dirigiremos al modelo de validación, donde buscaremos el Elemento adecuado de forma análoga a la expresada anteriormente en el apartado de construcción del modelo de datos.

Tras averiguar esta información y extraer el Elemento hijo, generaremos un nuevo nodo, del tipo NodoXML. Para configurarlo recurriremos al elemento extraído del modelo de validación. Los principales puntos a tener en cuenta en la configuración de un nuevo elemento como NodoXML son los siguientes:

- Nombre del elemento
- Creación del Element que referenciará para constituir el Document del objeto de aprendizaje en tiempo real
- Lista de posibles hijos del nodo: Cada uno indicando si es obligatorio y sus máximas apariciones.
- Lista de posibles Atributos del Nodo: Indicando si son obligatorios y sus valores por defecto.
- Flag que nos indica si el nodo es Simple, o Complejo, donde además puede resultar ser una Extensión.

Una vez generado, lo insertaremos en el padre gracias a la funcionalidad de Java ofrece en el trabajo con sus interfaces de nodos.

Gracias al modelo delegado de swing, esta modificación se verá automáticamente al refrescar la interfaz gráfica.

2. Eliminación de un elemento del objeto de aprendizaje

Para eliminar un nodo realizaremos un proceso sensiblemente más sencillo que el anterior.

En primer lugar debemos tener en cuenta que existen nodos obligatorios para un objeto de aprendizaje. Esta característica la determinará la gramática en la que esté basado, información que nosotros ya conocemos gracias al modelo de validación.

Lo primero que debemos hacer es acceder al modelo de datos. Una vez en él, identificaremos al padre. Tras analizar los diferentes hijos del padre y consultar en el modelo de validación la lista de hijos obligatorios del mismo, procederemos a eliminar este hijo.

Gracias, de nuevo, a las facilidades de Java para el trabajo sobre árboles, sencillamente utilizaremos estos métodos para eliminar el nodo que contiene el elemento del árbol. Será también eliminado del Document.

Esta información debe pasar al padre, que lleva un registro de hijos insertados y por insertar de cada tipo, actualizando estos contadores.

3. Modificación de un elemento del objeto de aprendizaje

Cualquier elemento del objeto de aprendizaje permite, mediante la interfaz, solicitar una modificación sobre el mismo, dado que previamente no conocemos las condiciones de este nodo. No obstante existen elementos que no poseen atributos modificables, y por tanto no podremos operar cambios sobre los mismos.

Ante la petición del usuario de modificación sobre un elemento que contiene atributos, el modelo de datos responderá de la forma que detallamos a continuación.

En primer lugar trataremos la información encontrada en las dos listas que este nodo contiene. La primera de ella acerca de todos los posibles atributos que el elemento puede contener, la segunda acerca de los atributos que actualmente posee, junto con sus valores.

Gráficamente se mostrará una lista completa con todos ellos, acompañados por sus valores aquellos que lo posean.

Ahora solo queda registrar la modificación sobre dichos valores que el usuario ha podido editar. Tras recoger la lista, recorreremos aquellos atributos que contienen valor no vacío, e introduciremos estos en el Element del NodoXML.

6.1.6 Interfaz

Como el resto de componentes software de la aplicación, la interfaz ha sido desarrollada bajo Java. Este lenguaje de programación ofrece un conjunto de librerías o paquetes destinados a la creación de interfaces de usuario interactivas muy completas.

Nuestra interfaz estará compuesta principalmente por elementos de los siguientes tipos:

- *JFrame*: Ventana principal de la aplicación, que contendrá al resto de elementos de la interfaz.

- *JPanel*: Paneles contenedores que facilitan la organización de la ventana principal.

- *JScrollBar*: Barras de desplazamiento para permitir al usuario desplazarse por paneles cuyo contenido excede los límites de su contenedor.

- *JToolBar*: Barra de herramientas donde se situarán los botones de la interfaz.

- *JButton*: Serán los botones de la aplicación. Permiten que se les añadan oyentes para los eventos que el usuario genere.

- *JMenu*: Constituirá la barra de menú principal de la aplicación, estructurada por funcionalidad. Pueden verse más detalles en la sección de especificación de requisitos.

- *JMenuItem*: Elemento del menú principal que, de forma análoga a los JButton, contendrá los oyentes para permitir al usuario acceder a todos los servicios de la herramienta.

- *JTree*: Elemento muy importante de nuestra interfaz, el cual representará la vista árbol del modelo de datos de la aplicación. Se sirve del modelo delegado de swing para construirse y acceder a dicho modelo.

- JEditorPane: Panel donde mostraremos el contenido de los recursos del objeto de aprendizaje.

- *JLabel*: Etiquetas de texto

- *JTextField*: Campos de texto editables para que el usuario introduzca los valores de los atributos.

Hasta aquí llegan los componentes gráficos que Java proporciona para la creación de interfaces, y que nosotros hemos utilizado. Vamos a ver ahora algunos detalles referentes al manejo de dicha interfaz.

6.1.6.1 Oyentes

Los oyentes son subclases de la ventana principal que implementan diversas interfaces de tratamiento de eventos. En nuestro caso trataremos principalmente con los eventos de pulsación de ratón.

Cada oyente se encarga de responder a los eventos generados por un grupo determinado de elementos de la interfaz. Vamos a ver los oyentes de los que disponemos en el siguiente fragmento de código extraído de la ventana principal de la aplicación.

```
/** Manejador para abrir objetos de aprendizaje */
private OpenLOListener openLOListener;

/** Manejador para guardar objetos de aprendizaje */
private SaveLOListener saveLOListener;

/** Manejador para comenzar nuevo objeto de aprendizaje*/
private NewLOListener newLOListener;

/** Manejador para abrir objetos QTI */
private OpenQTIListener openQTIListener;

/** Manejador para abandonar el programa */
private ExitFormListener exitFormListener;

/** Manejador para abandonar el programa */
private OyenteCerrarObjeto oyenteCerrarObjeto;

/** Manejador para el jeditorpane (browser) */
private BrowserListener browserListener;

/** Manejador para actualizar el jeditorpane (browser) */
private UpdateBrowserListener updateBrowserListener;

/** Manejador para cargar objetos en la BD */
private StoreDBListener storeDBListener;

/** Manejador para evaluar automaticamente objetos de aprendizaje
*/
private CBRListener cbrListener;

/** Manejador para realizar consultas sobre la base de datos de
objetos de           aprendizaje */
private QueryDBListener queryDBListener;

/** Manejador para componer objetos de aprendizaje en paralelo*/
private ComponerListener componerListener;
```

```
/** Manejador para componer objetos de aprendizaje en profundidad*/
private ComponerProfListener componerProfListener;

/** Manejador para guardar los objetos compuestos */
private GuardarComposicionListener guardarComposicionListener;

 /** Manejador para guardar todos los objetos compuestos */
 private GuardarTodoComposicionListener
 guardarTodoComposicionListener;

/** Manejador para salir de la composicion */
private SalirComposicionListener salirComposicionListener;

/** Manejador para mostrar objetos de aprendizaje */
private ShowLOListener showLOListener;

 /** Manejador para salvar los cambios en los ficheros del objeto
 de aprendizaje*/
 private GuardarCambiosListener guardarCambiosListener;

 /** Manejador que lanza la ventana de info corporativa */
 private InfoCorporativaListener infoCorpListener;

 /** Manejador que lanza la ventana de ayuda al usuario */
 private UserHelpListener userHelpListener;

 /** Manejador para gestionar el Arbol*/
 private TreeListener treeListener;

 /** Manejador para gestionar coordenadas del raton*/
 private OyenteMouse mouseListener;

 /** Manejador para el boton de modificacion*/
 private ModifyListener modifyListener;
```

6.1.6.2 Menús emergentes

Estos menús emergentes, conocidos también como Pop-up menú, nos servirán para realizar las tareas de edición de los objetos de aprendizaje.

Al igual que los *JMenu*, estos menús contienen un conjunto de JMenuItem que serán los encargados de agrupar la funcionalidad y generar los eventos con los cuales el usuario interactúa con el sistema, y en este caso, edita el objeto de aprendizaje.

6.1.6.3 Layout

Los layout serán las configuraciones o distribuciones que tendrán los componentes contenedores en la ventana. Cada Layout describe una diferente orientación que permite diseñar un entorno amigable y muy estético. Los layout utilizados en esta herramienta son los siguientes:

- *BorderLayout*: Divide la ventana en sectores Norte, Sur, Este, Oeste y Centro.

- *FlowLayout*: Este layout permite que los componentes se sitúen uno a continuación del otro. Es el layout por defecto que presenta Java.
- *GridLayout*: Permite dividir el componente en forma de rejillas.
- *BoxLayout*: Similar al anterior, pero distribuye cajas con una determinada orientación (Vertical u horizontal).

6.2 Módulo 2: Módulo de composición

Acorde con el resto del diseño, el módulo de composición respeta el Modelo-Vista-Controlador, sobre el que ya se habló anteriormente en la *Sección 5ª* de este documento, *Arquitectura del sistema*.

A continuación, daremos una visión general del empaquetamiento de objetos explicando el modelo de empaquetado de contenidos propuesto por la organización IMS denominado **"IMS Content Packaging Information Model" (IMS, 2004),** que describe la forma y estructura en que debe empaquetarse un objeto de aprendizaje, y en el que nos hemos basado para la realización de la composición de objetos.

Tras esto explicamos aspectos más técnicos sobre la implementación de este módulo en particular.

6.2.1 Introducción

La finalidad del empaquetamiento es agrupar contenidos relacionados entre sí de forma que sean tratados como una sola unidad y se pueda (Shih et al, 2003):

1) Estructurar los contenidos que forman el paquete.
2) Asociarle descripciones.
3) Facilitar su reutilización y que sean localizables en repositorios de contenidos.
4) Facilitar la agregación de distintos paquetes para forman paquetes más grandes.

6.2.1.1 Implicaciones en el empaquetamiento

Las acciones derivadas de llevar a cabo una composición están directamente influidas por la forma en que se va a implementar el empaquetamiento del objeto de aprendizaje. Por tanto, comentaremos las recomendaciones reflejadas en la especificación *IMS Content Packaging* (IMS, 2004) en las que nos hemos basado a la hora de implementar el módulo de composición para la herramienta LOMEditor:

- Modificación del manifiesto del objeto continente. Según la especificación *IMS Content Packaging*, todo objeto de aprendizaje debe tener un manifiesto principal que puede contener uno o más submanifiestos referidos a otros objetos de aprendizaje que son referenciados en el principal. Así pues la composición implica modificar el manifiesto del objeto continente en el siguiente sentido:

  1) Introducir el manifiesto del objeto contenido como submanifiesto del objeto continente.

2) Modificar la estructura lógica del objeto continente (expresada en el manifiesto a través de cualquiera de las organizaciones) para dar cabida a la referencia que se hace del objeto contenido.

3) Modificar los metadatos del objeto continente si la introducción del nuevo objeto en su estructura supone un cambio en el comportamiento del mismo.

4) Insertar el paquete físico del objeto contenido dentro de los recursos físicos que forman parte del paquete del objeto continente.

- La organización que aparece en un manifiesto representa una relación jerárquica en forma de árbol. Ésta se define en base a los elementos <ítem>, que actúan como nodos o hijo del árbol (por tanto, si tuviéramos un elemento <ítem> dentro de otro elemento <ítem>, representaría una relación padre-hijo). En este sentido existen diversas formas legales de referenciar submanifiestos o componentes de los mismos desde un manifiesto, usando para ello los elementos <ítem>.

- Referencias a submanifiestos. Sólo pueden referenciarse submanifiestos con una única organización que sean descendientes directos del manifiesto que contiene al ítem desde el que se les referencia. El resultado de la composición es la sustitución del ítem por un nuevo ítem que toma sus valores del nodo raíz de la organización del submanifiesto referenciado. En caso de que la organización no tuviera atributos que la identificaran, se mantendrían los atributos del ítem, y de éste colgarían los ítems de la organización no atribuida.
  Otro caso que podría darse, es que el ítem que referencia al submanifiesto contenga a su vez otros ítems. En este caso, el efecto de la composición consiste en añadir los ítems de la organización al mismo nivel que los ítems hijos del ítem que referencia al submanifiesto, y seguir las mismas reglas que para los casos anteriores acerca de los atributos del nuevo ítem.

- Referencias a recursos de un submanifiesto. Sólo pueden referenciarse recursos de submanifiestos que sean descendientes directos del manifiesto que contiene al ítem desde el que se les referencia. No es necesario que en el submanifiesto se haya definido una organización, pudiendo actuar como un simple contenedor de recursos.

- Con respecto a la modificación de los metadatos, se trata de una tarea principalmente manual sobre el manifiesto, que deberá realizar

el desarrollador del objeto a través de una herramienta de autoría. Esta tarea consiste en descubrir qué implicaciones tiene sobre el comportamiento y propiedades del objeto, la inserción del nuevo objeto (que en términos del manifiesto, consiste en modificar valores de las etiquetas del elemento metadata o la creación de nuevas etiquetas).

6.2.2 Composición de objetos de aprendizaje en la herramienta LOMEditor.

Componer dos objetos de aprendizaje significa obtener un único objeto a partir de los dos iniciales, y para ello es necesario (Shih et al, 2003):

- Definir la relación entre los objetos. La composición define una relación jerárquica entre los componentes siendo necesario establecer quién es el objeto contenido y quién es el objeto continente.

- Insertar el objeto contenido dentro del objeto continente. Esto implica llevar a cabo modificaciones sobre el objeto continente con respecto a la descripciones de su comportamiento, a los metadatos que lo describen y a la estructura lógica del mismo (observar que en el objeto contenido no es necesario llevar a cabo ninguna modificación).

6.2.2.1 Modelo interno de la aplicación

Información a representar por el modelo interno.

El elemento básico para llevar a cabo la composición es un modelo interno a la aplicación, en el que se representan los objetos de aprendizaje a componer.

La información a representar por el modelo interno de la aplicación en el módulo de composición, es similar a la representada en la funcionalidad de abrir objetos de aprendizaje (de la herramienta de autoría) explicada anteriormente. Sin embargo, ésta es algo más compleja debido a la creación del árbol de composición y los objetos de composición que forman el mismo.

Explicaremos el modelo interno de esta aplicación teniendo en cuenta los siguientes puntos:

El modelo de validación (o referencia) ha sido construido a partir de la gramática del objeto de aprendizaje (xsd del manifiesto y de metadata).

El manifiesto del objeto de aprendizaje ha sido validado mediante el modelo de validación anterior.

En base a estos dos puntos los elementos de los que consta el modelo fueron explicados en el apartado 6.1.3. Por tanto, explicaremos únicamente los nuevos elementos de los que consta el modelo interno para la creación del árbol de composición.

Los elementos explicados anteriormente aquí únicamente los nombraremos:

· *Modelo de validación.*

· *Document*.

· *Árbol temporal que representará el modelo.*

· *Árbol de composición que representará el modelo del árbol utilizado en la composición de objetos de aprendizaje:* Éste es el objeto principal en el modelo de la composición de objetos de aprendizaje y el que marca la diferencia con el modelo del árbol utilizado para abrir objetos de aprendizaje. Es un DefaultTreeModel. Se le añaden los objetos de composición.

· *Objeto de composición representará un objeto de aprendizaje:* Utilizado en la composición de objetos de aprendizaje. Creado a partir del nodo Organizations del mismo, que es un NodoXML del Arbol guardado anteriormente.
Así, para cada objeto de aprendizaje que mostremos (desde el menú Composición) para su posible composición con otro objeto, generaremos un *ObjetoComposicion*, que mostrará únicamente la información del nodo Organizations del objeto en cuestión.

Pasos para la construcción del modelo interno de la aplicación.

Los pasos explicados anteriormente únicamente los enumeramos:

1. Se construye un objeto del tipo Arbol (ver clase Arbol).
2. Generación del Árbol Java.
3. Construcción del NodoXML (ver clase NodoXML).
4. Organización del árbol (DefaultTreeModel) de la clase Arbol
5. Creación del objeto árbol de composición (ver clase *ArbolComposicion*)
    a. Creamos el modelo del árbol compuesto.

b. Creamos el nodo raíz del mismo.

6. Construcción del objeto de composición (ver clase ObjetoComposicion)

   a. La clase ObjetoComposicion extiende de DefaultMutableTreeNode, con lo que podemos aprovechar todos los métodos de trabajo sobre nodos y árboles.

   b. Guardamos la información necesaria de este nodo:

      i. Nombre del elemento.
      ii. Nodo Organizations, que lo obtendremos del modelo del Árbol de este objeto de aprendizaje, generado anteriormente.

7. Organización del árbol (DefaultTreeModel) de la clase ArbolComposicion.

   a. Se inserta en el árbol de composición el nodo Organizations del objeto de aprendizaje actual.

8. Construcción final del JTree:

6.2.2.2 Detalles técnicos

Para realizar la composición de dos objetos de aprendizaje, previamente el árbol de composición deberá contener dos ó más objetos de composición. Denominamos a esta acción *mostrar objeto de aprendizaje* y no *abrir objeto de aprendizaje,* puesto que el objeto de aprendizaje no es mostrado completamente en el árbol de composición, ni se permite la visualización o edición de su contenido. Este árbol de composición representará el modelo del árbol utilizado en la composición de objetos de aprendizaje. Es un objeto de tipo *ArbolCompuesto*. Está formado por un DefaultTreeModel, que es el modelo del árbol para la construcción del JTree, y por el nodo raíz del árbol, de tipo DefaultMutableTreeNode, al que se añadirán todos los objetos de composición que se vayan creando a partir de cada objeto de aprendizaje que se quiera mostrar en el mismo.

Al realizar la acción de *mostrar un objeto de aprendizaje* se crea lo que hemos denominado *objeto de composición*. Este objeto es del tipo ObjetoComposicion y extiende de la clase DefaultMutableTreeNode. Al constructor se le pasa el nombre del objeto de aprendizaje que se quiere mostrar y un NodoXML.

El objeto se crea de la siguiente forma:

- Se crea un DefaultMutableTreeNode con el nombre del objeto de aprendizaje.

- Se le añade el NodoXML, que es el nodo Organizations del objeto de aprendizaje que se quiere mostrar, y que lo obtenemos del Arbol generado previamente a partir de este objeto de aprendizaje.

Ningún objeto es mostrado en el árbol de composición sin haber sido validado previamente. (Ver en el apartado 6.2 la explicación de la validación).

Una vez almacenados un mínimo de dos objetos de composición en la lista *ListaObjAprendizaje*, podremos componer estos objetos. La clase ListaObjAprendizaje contiene un *ArrayList* de objetos de tipo *ObjAprendizaje*. Cada uno de estos objetos representa un objeto de aprendizaje y almacena los datos necesarios para realizar la composición. Los objetos ObjAprendizaje se irán eliminando de la lista según sean utilizados como objetos hijos de la composición con otro objeto.

La composición entre dos objetos de aprendizaje puede realizarse de dos maneras diferentes: mediante la *composición en paralelo*, en la que el objeto de aprendizaje que actúa como contenido será un hijo directo de la raíz de una organización del objeto que actúa como continente; y la *composición en profundidad*, en la que el objeto que actúa como contenido va a ser un hijo de algún hijo interno de la raíz de una organización del objeto que actúa como continente.

Composición en paralelo

Para la composición en paralelo el objeto de aprendizaje que actúa como contenido será un hijo directo de la raíz de una organización del objeto que actúa como continente.

Como ya hemos explicado anteriormente, es un requisito imprescindible para la composición tener un mínimo de dos objetos de aprendizaje almacenados en la lista de la clase ListaObjAprendizaje, ya que la composición se realiza entre dos objetos de composición.

La manera de realizar esta composición es la siguiente: en la clase Composición se recorre el manifiesto del objeto contenedor buscando la organización que será padre en el objeto compuesto. Ya que esta organization sólo será hijo directo de la raíz de una organización del objeto que actúa como continente, únicamente será necesario recorrer los hijos del manifest hasta encontrar el hijo organizations y, una vez encontrado, se recorrerán sus hijos,

que son elementos organization, por tanto entre ellos se encontrará el elemento organization buscado.

Una vez encontrada esta organización, se recorre el manifiesto del objeto contenido, buscando la organización que será hija en el objeto compuesto. A partir de esta segunda organización se crea un ítem para que pueda ser añadido en último lugar como hijo directo de la organización encontrada anteriormente en el objeto padre.

Una vez añadido dicho ítem al manifiesto padre, se añadirá el manifest del objeto contenido como submanifest en el objeto contenedor, de forma que sea un hijo directo del manifiesto de este objeto.

Composición en profundidad

En la composición en profundidad el objeto que actúa como contenido va a ser hijo de algún hijo interno de la raíz de una organización del objeto continente.

Para ello, al igual que en la composición en paralelo, es imprescindible la existencia de al menos dos objetos de aprendizaje en la lista de la clase ListaObjAprendizaje.

Para la composición en profundidad, también se recorrerá el manifiesto del objeto contenedor en la clase Composición, pero esta vez buscando el ítem que actuará como padre en el objeto compuesto. Para ello, ya que este ítem puede ser cualquier hijo interno de la raíz de una organización, se deberán recorrer los hijos del manifest hasta encontrar el hijo organizations. Una vez encontrado, se recorrerán sus hijos, que son elementos organization, y para cada uno se recorrerán sus hijos y así recursivamente hasta encontrar el ítem seleccionado.

Una vez localizado el ítem padre en el objeto contenedor, se recorrerá el manifiesto del objeto contenido, buscando la organización que será hija en el objeto compuesto, al igual que se hacía en la composición en paralelo. Con esta organización se creará un ítem para que pueda ser añadido en último lugar como hijo directo del ítem localizado anteriormente en el objeto padre.

Tras ser añadido dicho ítem al ítem del manifiesto padre, se añadirá el manifest del objeto contenido como submanifest en el objeto contenedor, de forma que sea un hijo directo del manifiesto de este objeto.

Para ambas composiciones, estos cambios en el manifest del objeto contenedor quedarán reflejados en el documento xml (*imsmanifest.xml*) del mismo, pero únicamente en el manifiesto descomprimido en la carpeta temporal de este objeto.

Los cambios realizados durante la composición son guardados únicamente en una carpeta temporal que es creada al inicio de la composición (al mostrar objetos de aprendizaje) y es eliminada al finalizar la misma (en el caso del objeto contenido, la carpeta temporal se elimina al finalizar su composición. En el caso del objeto continente, la carpeta temporal se eliminará al salir de la composición). Si los cambios realizados no han sido guardados, es decir, si el objeto compuesto no ha sido comprimido previamente (mediante las funcionalidades Guardar como... y Guardar todo), al salir de la composición y ser eliminadas dichas carpetas, los cambios realizados serán eliminadas con ellas.

Para escribir los cambios en el documento utilizamos las clases *FileOutputStream* y *xmlOutputter*. Esto está implementado en la clase *Composicion* (en la función *modificarManifiestoPadre(...)*) de la siguiente forma:

```
try
{
    //Escribo en el xml

    FileOutputStream fos = new FileOutputStream(path + File.separator
    + "imsmanifest.xml");

    xmlOutputter.setFormat(Format.getPrettyFormat());
    xmlOutputter.setFormat(xmlOutputter.getFormat().setEncoding("ISO-
    8859-1"));

    xmlOutputter.output(manifiestoPadre, fos);
}
catch (IOException ioe)
{
    System.out.println("Ha ocurrido un error al intentar escribir en
    el documento." + ioe);
}
```

Para completar la composición de estos objetos es necesario copiar los recursos (únicamente los recursos) del objeto contenido al objeto contenedor. A continuación se muestra el código implementado:

```
/** Método que copia los archivos necesarios del objeto hijo al
objeto padre
*/
public void copiarArchivos()
{
       String rutaOriginal = controller.getRutaTemp(numDocHijo);
       //Elimino del hijo los archivos que no quiero copiar en el
       //padre (los que no son recursos)
       File carpeta = new File(rutaOriginal);
```

```
        File fileManifest=new File(rutaOriginal+"\\imsmanifest.xml");
        if (fileManifest.exists())  fileManifest.delete();
        File    fileXSDManifest    =    new    File(rutaOriginal    +
        "\\imscp_v1p1.xsd");
        if (fileXSDManifest.exists())  fileXSDManifest.delete();
        File    fileXSDMetadata    =    new    File(rutaOriginal    +
        "\\imsmd_v1p2p2.xsd");
        if (fileXSDMetadata.exists())  fileXSDMetadata.delete();

        File[] files = carpeta.listFiles();

        //Copiamos los recursos del hijo a padre
        for(int i=0;i<files.length;i++)
        {
              String                    rutaDestino                    =
              controller.getRutaTemp(numDocSelected);
              String ruta = files[i].getPath();
              int index = ruta.lastIndexOf(File.separator);
              String archivo = ruta.substring(index, ruta.length());
              rutaDestino = rutaDestino + archivo;
              if(!files[i].isDirectory())
              {
                    copiarUnArchivo(ruta, rutaDestino);
              }
              else
              {
                    File carpetaNueva = new File(rutaDestino);
                    carpetaNueva.mkdir();
              }
        }
}



/** Método que copia un archivo localizado en la rutaOriginal dada
en la rutaDestino
*/
public void copiarUnArchivo(String rutaOriginal, String rutaDestino)
{
      try
      {
            FileInputStream    fis    =    new    FileInputStream(new
            File(rutaOriginal));
            FileOutputStream   fos   =   new   FileOutputStream(new
            File(rutaDestino));

            byte[] buf = new byte[1024];

            int i = 0;
            while ( (i = fis.read(buf)) != -1)
            {
                  fos.write(buf, 0, i);
            }
            fis.close();
            fos.close();
            rutaOriginal = rutaDestino;
      }
      catch(IOException ioe)
      {
            System.err.println("Error  al  copiar  los  recursos\n" +
            ioe);
      }
```

```
}
```

Una vez realizada la copia de recursos en el objeto utilizado como padre en la composición, se actualiza la lista ListaObjAprendizaje eliminando de la misma el objeto utilizado como hijo en la composición.

Se elimina además la carpeta temporal de este objeto, tal como se explicó anteriormente.

Por tanto, la carpeta temporal del objeto padre ha sido modificada de forma que ahora contiene el objeto compuesto en lugar del objeto de aprendizaje abierto inicialmente.

Para finalizar correctamente la composición y tener realmente un objeto compuesto, es necesario que esta carpeta temporal sea comprimida (para ello es necesario elegir desde la interfaz una de las funcionalidades para zipear el objeto compuesto, que son Guardar objeto compuesto como... o Guardar todo). Para ello se hará uso de la clase *ZipOutputStream*, con lo que podremos comprimir el archivo en formato *zip*. Se hará una llamada a esta función, ya que está implementada fuera del paquete de composición.

6.3 Módulo 3: Módulo de evaluación de calidad

Como el resto del diseño, este módulo respeta totalmente el Modelo-Vista-Controlador, sobre el que ya se habló en la *Sección 5ª* de este documento, *Arquitectura del sistema*.

A continuación, por tanto, se desarrollan aspectos más técnicos sobre la implementación de este módulo en particular.

6.3.1 Introducción

La herramienta LOMEditor 2005 permite realizar la evaluación de objetos de aprendizaje desde un punto de vista general, es decir, desde una visión superficial, la que ofrece la parte de los metadatos generales del objeto de aprendizaje.

Se pueden evaluar los objetos de aprendizaje de dos maneras diferentes: mediante la opción de *evaluación manual,* en la que el usuario es el encargado de evaluar las distintas etiquetas del objeto de aprendizaje, y *evaluación automática,* en la que mediante las técnicas relatadas a continuación, basadas en el razonamiento basado en caso, se asocia al objeto de aprendizaje una evaluación sin que el usuario tenga que realizar ningún esfuerzo.

6.3.2 Evaluación manual

Esta opción concede al usuario la posibilidad de introducir un objeto de aprendizaje en la base de datos de la herramienta, que además servirá como base de casos del sistema *CBR* (*Cased Based Reasoning*) para la evaluación automática que más adelante comentaremos.

Para poder guardar la estructura del objeto de aprendizaje y de sus metadatos es necesario realizar un parsing o mapeado del archivo *imsmanifest.xml.* LOMEditor 2005 realiza este parsing en dos pasos: primero, se almacena la estructura del objeto en clases, diseñadas para este propósito, y seguidamente, se almacena el contenido de las clases anteriores en la base de datos del sistema.

6.3.2.1 Parsing del XML

Por cada elemento del archivo *imsmanifest.xml* existe una clase encargada de almacenar toda la información relacionada con ese elemento. La nomenclatura elegida para cada una de estas clases es la siguiente: *nombre_del_elemento+Container,* por ejemplo, *MetadataContainer.* Asimismo, existe una clase que se encarga de almacenar a todas las anteriores, así como otros

objetos relacionados con la funcionalidad de evaluar objetos de aprendizaje. Esta clase se llama *ModeloCBR*.

El mapeo del manifiesto lo lleva a cabo la clase *ModeloCBR*, la cual recorre mediante JDOM el árbol del manifiesto y, por cada elemento, almacena la información asociada a dicho elemento en las clases *Container*. Esta clase es la encargada de soportar todas y cada una de las acciones que se realizan sobre el modelo de la aplicación y que tienen que ver con aspectos relacionados con la evaluación de calidad de los objetos de aprendizaje.

En relación al mapeo del XML, la clase *ModeloCBR* se encarga de rellenar la estructura que forman las denominadas clases *Container,* guardando toda la información necesaria del XML del manifiesto del objeto de aprendizaje en estas clases. Como la evaluación de objetos de aprendizaje, sólo se basará en el contenido de las etiquetas referentes a los metadatos del objeto de aprendizaje, etiquetas hijas de <metadata> según *IMS Metadata Specification*, la información que almacenarán estas clases sólo será la de estas etiquetas. *Véase el diagrama A.1.3.1 del Apéndice A.*

Actualmente, sólo se mapea la parte del manifiesto correspondiente a los metadatos de tipo general, es decir, a los que se incluyen en la etiqueta <general>, que a su vez se encuentra dentro de la etiqueta <lom>, que es la etiqueta raíz de los metadatos. En un futuro no muy lejano está prevista su ampliación para que se mapeen también los metadatos de las etiquetas <educational> y <technical>, hijas también de la etiqueta <lom>. Esta ampliación dotaría al sistema CBR de una potencia y fiabilidad mucho mayores de las actuales ya que las técnicas de similitud entre objetos de aprendizaje serían más exactas que las actuales. No obstante, la intención de este proyecto era sólo la de vislumbrar unos de los posibles caminos a seguir en el campo de la evaluación de objetos de aprendizaje, por lo que un prototipo de juguete que muestra lo que se podría llegar a conseguir con este tipo de técnicas es suficiente.

6.3.2.2 Evaluación del objeto de aprendizaje

Una vez almacenada la estructura del objeto de aprendizaje en memoria dinámica, mediante las clases *Container,* el siguiente paso es la evaluación a mano, de las etiquetas que posee el objeto de aprendizaje, correspondientes a los metadatos de tipo general.

Este objetivo se debe alcanzar con ayuda del usuario, y a estos efectos, se utilizarán las clases container del *ModeloCBR* para generar una ventana dinámicamente, en la que se le muestre al usuario la posibilidad de evaluar cada etiqueta de los metadatos que el sistema previamente ha evaluado, tal como se explica en el paso anterior. La clase *VentanaEvalGeneral* será la encargada de mostrar este formulario solicitando la evaluación del usuario para cada etiqueta de la

que se guardó información en el *ModeloCBR*, con un barómetro de 0 a 4, que se presentará al usuario como desde Muy bueno a Muy malo, atendiendo a la siguiente tabla de correspondencias:

Muy bueno ----- 4
Bueno ----- 3
Regular ----- 2
Malo ----- 1
Muy malo ----- 0

Tras haber evaluado las etiquetas y pulsado aceptar, LOMEditor realiza una evaluación desde un punto de vista general del objeto de aprendizaje atendiendo a los valores introducidos por el usuario para cada etiqueta. Esta evaluación la lleva a cabo la clase *EstrategiasEvaluacion,* que también está incluida en la clase *ModeloCBR.* Esta clase recoge las estrategias de evaluación de aprendizaje posibles en el sistema. Actualmente, se decidió que la estrategia de evaluación a seguir para obtener la evaluación global del objeto de aprendizaje fuese la media aritmética de los valores introducidos por el usuario. No obstante, dotando a esta clase de otras métricas más específicas se puede aumentar también la precisión en la obtención de la evaluación de calidad.

6.3.2.3 Almacenamiento en la base de datos del sistema

El último paso en la evaluación manual será el de guardar el objeto de aprendizaje, así como su evaluación asociada, en la base de datos de LOMEditor 2005.

Este cometido lo llevan a cabo las clases *Container*, las cuales son el único punto de acceso a la base de datos de la herramienta. De este modo, la clase *GeneralContainer,* en la cual se encuentra almacenada toda la información asociada a la etiqueta <general> de los metadatos del objeto de aprendizaje, se encarga de guardar cada elemento que la constituye, recorriendo la estructura de datos que lo representa e insertando su contenido en la tabla correspondiente de la base de datos.

De esta manera, se rellena el contenido de cada una de las tablas que se representan en el *Diagrama A.3.2 del Apéndice A*, sobre las que se tiene información en el *ModeloCBR*.

Nota: Sólo se rellenan las tablas correspondientes a los elementos hijos de la etiqueta <general> de los metadatos de cada objeto de aprendizaje, así como la evaluación asociada a cada objeto (sólo la evaluación de tipo general, por el momento). El resto de tablas serían las necesarias para realizar evaluaciones desde el punto de vista pedagógico.

6.3.3 Evaluación automática o evaluación basada en CBR

Esta segunda funcionalidad de LOMEditor permite al usuario evaluar, de manera rápida y sin apenas esfuerzo, un objeto de aprendizaje mediante una estrategia de CBR sobre la bbdd del sistema.

Los pasos que se siguen en la evaluación de objetos mediante CBR son los siguientes:

1. Se realiza un parsing del objeto de aprendizaje y se guarda su contenido en las clases Container, tal y como se ha explicado anteriormente que se hacía cuando el usuario introducía un objeto en la bbdd.

2. Se van extrayendo, uno por uno, los objetos de la bbdd, que servirán como *casos de prueba* en la evaluación CBR.

3. Por cada caso de prueba extraído de bbdd, se realizará una *estimación de similitud* con respecto al objeto que estamos evaluando, y que nos proporcionará un valor de similitud entre el caso de prueba y dicho objeto.

4. Una vez evaluados todos los casos de prueba que hay almacenados en la bbdd, y sabiendo ya cual de todos es el objeto más similar al que estamos intentando evaluar, se toma la evaluación de ese objeto y se asigna al actual.

5. Se muestra al usuario un formulario en el que puede elegir entre: aceptar y que tanto el objeto como su evaluación se guarden en la base de datos (aprendizaje automático), o cambiar de manera manual la evaluación generada automáticamente por el sistema mediante el CBR.

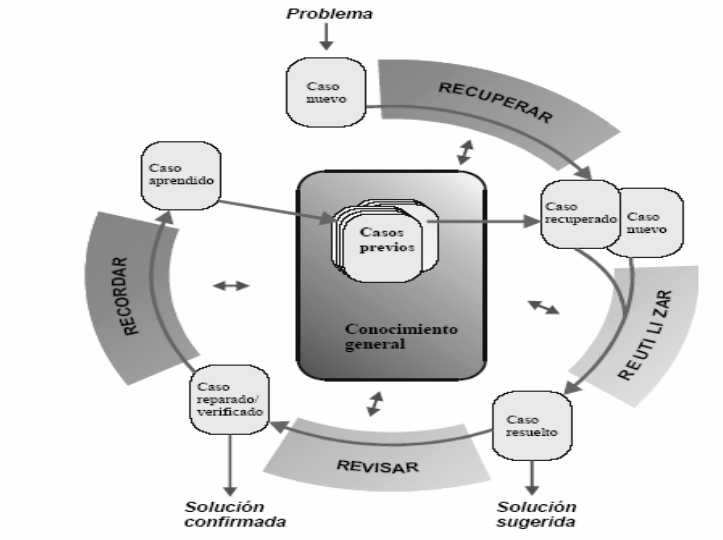


Imagen 1: Esquema de actuación de sistemas CBR

6.3.3.1 Base de casos del sistema CBR

La base de casos de nuestro sistema CBR es el motor que proporciona toda la potencia a esta técnica.

LOMEditor 2005 implementa su base de casos mediante una bbdd relacional en la cual va guardando todos los casos de prueba del sistema. El soporte que utiliza LOMEditor para su base de datos es MySQL®.

Cada caso se guarda en la base de casos en diferentes tablas, pudiendo ser recompuesto cada objeto gracias al campo IDENTIFIER que se guarda en todas las tablas de la base de datos y que hace referencia al identificador del manifiesto que representa la estructura del objeto de aprendizaje, y que es único para cada objeto.

6.3.3.2 Extracción de casos de la base de datos

La extracción de casos de la base de datos supone una recomposición de la estructura del manifiesto del objeto, necesaria para su almacenamiento en la clase *MetadataContainer,* la cual será la encargada de realizar la evaluación de similitud entre el objeto a evaluar y cada uno de los casos que contiene la base de datos del sistema. Esta recomposición, como se indicaba anteriormente, se basa en el campo IDENTIFIER, existente en cada una de las tablas de la base de datos, y que indica a qué objeto único pertenece la información del registro asociado a cada identificador de manifiesto.

6.3.3.3 Evaluación de similitud

Para llevar a cabo la evaluación basada en CBR el sistema realiza una evaluación de similitud entre cada uno de los casos que se encuentran en la base de casos del sistema y el caso actual a evaluar. La encargada de realizar esta evaluación es la clase *ModeloCBR*, la cual a partir de dos objetos de la clase *MetadataContainer* establece un resultado de comparación entre ambos, que será un número real entre 0 y 4, y que estará basado en las técnicas de similitud que se explican en el siguiente apartado de este documento.

El método de evaluación de similitud entre dos objetos de aprendizaje tendría el siguiente aspecto:

```
/** Método que evalúa automáticamente el objeto de aprendizaje actual
   * utilizando la técnica de Inteligencia Artificial llamada CBR
   */
  public Evaluacion evaluacionCBR()
  {
    int masParecido = -1;
    double maxComp = 0.0;
    ArrayList ids = metadata.getIDManifiestos();
    if (ids != null)
    {
      for (int i = 0; i < ids.size(); i++)
      {
        MetadataContainer mc = cargaOABD( (String) ids.get(i));
        double comparacion = compara(metadata, mc);
        if (maxComp <= comparacion)
        {
          maxComp = comparacion;
          masParecido = i;
        }
      }
      //Hay un objeto que se parece al actual
      if (masParecido != -1)
      {
        return cargaEvaluacionBD( (String) ids.get(masParecido));
      }
      //Todos los objetos de la BD son totalmente diferentes al actual
      else
      {
        return null;
      }
    } //if(ids !=null)
    return null;
  }
```

6.3.3.4 Técnicas de similitud

Las técnicas similitud que utiliza la herramienta LOMEditor 2005 son muy básicas, pero demuestran que, aún de manera tan básica, el potencial que tienen las técnicas de CBR en la evaluación de objetos de aprendizaje es muy grande y ampliando y mejorando estas técnicas la evaluación de objetos podría legar a ser muy precisa.

Para cada una de las etiquetas de hijas de <general> de metadata se realiza un encaje de patrones exacto, es decir, se compara si el contenido de cada etiqueta es exactamente el mismo en los dos objetos que se están comparando.

Las etiquetas que se comparan y la forma de hacerlo es la siguiente:

<catalogentry> → se compara el contenido del elemento <catalog>, si los dos objetos analizados contienen un elemento <catalog> con el mismo contenido se da una evaluación de 1, 0 en caso contrario. Como puede haber varios elementos de tipo <catalogentry> en un mismo objeto de aprendizaje se hace una media de todos los resultados obtenidos.

<language> → se compara el contenido del elemento, si los dos objetos analizados contienen un elemento <language> con el mismo contenido se da una evaluación de 1, 0 en caso contrario. Como puede haber varios elementos de tipo <language> en un mismo objeto de aprendizaje se hace una media de todos los resultados obtenidos.

<keyword> → se compara el contenido del elemento, si los dos objetos analizados contienen un elemento <keyword> con el mismo contenido se da una evaluación de 1, 0 en caso contrario. Como puede haber varios elementos de tipo <keyword> en un mismo objeto de aprendizaje se hace una media de todos los resultados obtenidos.

<structure> → se compara el contenido del elemento <value>, si los dos objetos analizados contienen un elemento <value> con el mismo contenido se da una evaluación de 1, 0 en caso contrario.

<aggregationlevel> → se compara el contenido del elemento <value>, si los dos objetos analizados contienen un elemento <value> con el mismo contenido se da una evaluación de 1, 0 en caso contrario.

Como se puede observar este es otro de los puntos en los que se podría aumentar en muchos puntos la potencia del sistema CBR. Técnicas de similitud que tengan en cuenta la importancia de las diferentes etiquetas y den pesos a éstas a lo hora de evaluar la similitud entre dos objetos conseguirían unos resultados más precisos. Además, se podría mejorar el modo en que se comparan las etiquetas introduciendo en el sistema técnicas de análisis del lenguaje natural, o basadas en árboles semánticos u ontologías. No obstante, todas estas técnicas atravesaban nuestra intención de mostrar en una aplicación "de juguete" los resultados que se podrían obtener al aplicar las técnicas de los sistemas expertos basados en CBR a la evaluación de los objetos de aprendizaje.

6.3.3.5 Aprendizaje automático

LOMEditor 2005 es capaz de aumentar de manera automática su base de casos. Así, tras la evaluación automática de cada objeto de aprendizaje, se mostrará una ventana con información al usuario del resultado obtenido. Si el usuario considera que la evaluación es correcta el sistema almacenará el objeto evaluado y su evaluación obtenida de manera automática, aumentando su base de casos para una mayor exactitud en posteriores evaluaciones.

6.3.4 Restricciones del CBR

La potencia de la evaluación automática que ofrece LOMEditor está supeditada a la cantidad de metadatos que contiene el objeto de aprendizaje. Si el objeto de aprendizaje evaluado no tiene o tiene rellenos muy pocos metadatos la evaluación del objeto no se ajustará demasiado a la realidad, ya que la comparación con los casos de prueba de la base de casos será muy pobre.

Además, la evaluación CBR de los objetos mejora a manera que crece la base de casos del sistema, es decir, a manera que el sistema aprende, debido a que al contener más casos de prueba sobre los que realizar comparaciones hay una mayor probabilidad de que haya objetos más similares a los que evaluamos.

6.3.5 Ventajas del CBR

El sistema CBR actúa de manera similar a como lo hace la mente humana en la búsqueda de una solución para un problema dado. El sistema busca en su memoria (base de casos) un problema (caso) lo más similar posible al problema al que se enfrenta en este momento y ofrece una solución similar a la que ofreció en el caso más similar.

La adquisición de conocimiento no supone un cuello de botella, como en otro tipo de técnicas como en los RBR (Rule Based Reasoning).

La capacidad de aprendizaje permite que el sistema aprenda de manera automática y mejore su precisión a la hora de evaluar objetos de aprendizaje.

SECCIÓN 7ª

CONCLUSIONES Y TRABAJO FUTURO

7.1 Conclusiones

7.1.1 Situación final del proyecto

Tras el proceso de trabajo realizado para este proyecto, hablaremos de la situación en la que actualmente ha quedado el mismo.

Finalmente ha sido posible que la herramienta de autoría de objetos de aprendizaje LOMEditor 2005 sea realidad, actualmente en forma de prototipo beta.

Toda la funcionalidad prevista en la especificación de requisitos y planificación de alcance ha podido ser desarrollada. Esta funcionalidad, adecuadamente probada, es la siguiente:

1.- Módulo principal de la herramienta de autoría

Este módulo es el núcleo de la aplicación. Comprende la funcionalidad general de trabajo con objetos de aprendizaje.

La funcionalidad concreta es la siguiente:

- Apertura de objeto de aprendizaje previa validación, que incluye la visualización del contenido del objeto de aprendizaje.

- Edición de un objeto de aprendizaje, que comprende la inserción de nuevos elementos, la eliminación de los mismos y de la posibilidad de modificar sus atributos.

- Salvar el manifiesto (imsmanifest.xml) del objeto de aprendizaje con el que se está trabajando.

- Guardar un objeto de aprendizaje abierto en forma comprimida.

- Creación de un nuevo objeto de aprendizaje, partiendo de cero.

- Ayudas

- Consultas sobre una base de datos que contiene información de objetos de aprendizaje.

2.- Módulo de composición

Este módulo es el encargado de implementar los servicios que LOMEditor ofrece a sus usuarios para que estos puedan componer objetos de aprendizaje a

partir de objetos anteriormente generados. Su funcionalidad está completamente desarrollada, aunque no se descartan futuras ampliaciones.

La funcionalidad concreta es la siguiente:

- Mostrar objetos de aprendizaje previa validación.

- Composición en paralelo de dos objetos de aprendizaje.

- Composición en profundidad de dos objetos de aprendizaje.

- Guardar objetos compuestos en forma comprimida.

- Guardar todos los objetos compuestos hasta el momento.

- Salir de la composición.

3.- Módulo de evaluación de la calidad

Este módulo comprende la funcionalidad que la herramienta dedica a la categorización de objetos de aprendizaje ya creados. Es el único módulo no desarrollado completamente, del que se dispone tan solo de un primer prototipo.

La funcionalidad concreta, en estos momentos, es la siguiente:

- Evaluación manual de la calidad de objetos de aprendizaje abiertos y almacenamiento persistente en una base de datos.

- Clasificación automática de objetos de aprendizaje de acuerdo a los mismos atributos usados en su evaluación.

7.1.2 Trabajo realizado y conocimientos adquiridos

No queremos cerrar esta memoria sin realizar un pequeño análisis o reflexión acerca del trabajo realizado por los componentes del grupo.

Introduciremos primero una breve descripción del proceso de desarrollo y metodología seguida.

Para el desarrollo de este proyecto ha sido necesario un complejo trabajo de coordinación y organización, donde las nociones adquiridas en Ingeniería del Software han resultado determinantes.

En primer lugar se ha realizado una especificación de los puntos principales a desarrollar, junto con una declaración de intenciones o alcance. Una vez establecidos estos puntos, se han decidido la arquitectura y el diseño en el cual deberíamos basarnos, aspectos a los cuales hemos procurado ser muy fieles, ya que de ello dependía la buena conclusión de LOMEditor.

Como se ha referido en otras secciones, la arquitectura elegida proporcionaba un fácil reparto del trabajo, que nos ha permitido gestionar correctamente nuestro esfuerzo. La constante comunicación de los miembros del grupo, ya fuese mediante reuniones formales o mensajería electrónica, también ha sido un factor determinante a la hora de garantizar la correcta marcha del trabajo.

Finalmente cada integrante del equipo de desarrollo ha cumplido satisfactoriamente con el trabajo que tenía asignado y LOMEditor se ha convertido en una realidad.

Todo este proceso nos ha llevado a la adquisición de ciertos conocimientos verdaderamente valiosos dentro de nuestro sector. Lo más destacable dentro del amplio conjunto de información asimilada podríamos decir que es lo siguiente:

1.- Nociones sobre e-learning

Hasta la realización de este proyecto nuestros conocimientos generalizados, salvo alguna excepción, eran ciertamente escasos. Tras desarrollar una herramienta directamente integrada en este campo poseemos una visión mucho más completa y precisa de todo ello.

2.- Estándares y especificaciones

En sectores de trabajo tan amplios como el que nos concierne, donde la evolución a nivel internacional es tan constante y marcada, se hace muy necesaria la utilización de estos estándares y especificaciones. La experiencia adquirida en la interpretación e implementación de dichos patrones es un punto muy destacable del conocimiento que, gracias al proyecto, ahora poseemos.

3.- Investigación y aprendizaje de nuevas tecnologías

Dado que muchos aspectos de esta herramienta son totalmente innovadores, ha sido necesario un proceso de investigación y aprendizaje de nuevas tecnologías hasta encontrar aquellas que verdaderamente han podido resultarnos útiles.

4.- Trabajo sobre J2SE

La necesidad de una plataforma que ofreciese un potente conjunto de facilidades nos ha llevado a elegir Java como base de nuestra herramienta. Actualmente es el lenguaje de referencia en las aplicaciones de desarrollo de estas características, y posee además un conjunto de paquetes de utilería verdaderamente valioso para satisfacer nuestras expectativas.

Dado que hemos tenido que servirnos no solo del paradigma de programación orientado a objetos que Java fomenta, sino también de ciertos detalles de diseño y arquitectura que esta plataforma contribuye a mejorar, podemos decir que nuestra experiencia sobre esta plataforma se ha visto sensiblemente incrementada.

5.- Utilización de tecnologías de marcado y organización

Dado que la información principal del objeto de aprendizaje se encuentra en un manifiesto de tipo XML, y la gramática en la que se basa es XSD, nos vimos en la necesidad de profundizar en la aplicación de estas tecnologías. También, para la visualización del contenido del objeto de aprendizaje, recurrimos a HTML y hojas de estilo.

Este conocimiento, teniendo en cuenta el fuerte impacto de dicha tecnología actualmente, supone un valor muy a tener en cuenta.

6.- Experiencia práctica en desarrollos de gran volumen

De nuevo hay que hacer especial mención a los conocimientos de Ingeniería del Software que hemos tenido oportunidad de aplicar para desarrollar esta herramienta, que supone un proyecto de gran volumen, considerando el alcance del mismo. Dentro de esta experiencia incluimos la especificación del proyecto, el trabajo coordinado en grupo y la obtención final de un producto satisfactorio.

En conclusión, ha sido posible la creación de un producto que consideramos francamente satisfactorio, como es la herramienta de autoría de objetos de aprendizaje LOMEditor. A nivel personal, destacamos la satisfacción que nos ha producido ver hacerse realidad un proyecto bastante ambicioso que trata de innovar y mejorar la oferta actual.

7.2 Trabajo futuro

El proyecto realizado puede ser extendido en cuatro sentidos que a continuación se comentan:

7.2.1 Personalización y aprendizaje adaptativo

Uno de los objetivos buscados por un sistema enseñanza basado en computador es la personalización del aprendizaje a las características particulares de cada alumno, y la adaptación al progreso que tiene el mismo en su aprendizaje. Así un sistema que permitiera personalización y adaptación debería ser capaz de:

a. Extraer un conjunto de rasgos definitorios del nivel y situación del alumno, y representativos para el tipo de aprendizaje que se va a realizar. La obtención de los rasgos puede ser directa como por ejemplo rellenando un formulario, o bien indirecta mediante un seguimiento de la forma de interactuar del alumno con el sistema.
b. A continuación se debe llevar a cabo una selección de los contenidos y la forma de presentarlos acorde a los rasgos extraídos. La asociación existente entre rasgos y contenidos vendrá definida a partir de reglas de definición introducidas por el implementador del sistema, y que podrían estar basadas en algún modelo pedagógico.
c. En último lugar se debe realizar la presentación del aprendizaje, la cual conllevará sucesivas selecciones sobre los contenidos seleccionados inicialmente de acuerdo a las respuestas y retroalimentación que el sistema sea capaz de obtener del alumno.

La puesta en práctica de un sistema con estas características hace necesario una manipulación eficiente de los contenidos, lo que implica contenidos suficientemente estructurados y categorizados sobre los cuales poder realizar búsquedas y procesamientos.

En el ámbito de los objetos de aprendizaje, disponer de personalización y adaptación hace necesario que el marcado de los objetos de aprendizaje tenga un nivel de granularidad suficientemente fino, y permite una aproximación adaptativa del aprendizaje basada en asociar los metadatos del objeto con las características individuales del alumno o a los propósitos de las organizaciones que quieren usarlo.

7.2.2 Disponer de un método para buscar y recuperar contenidos

La búsqueda y recuperación de contenidos se debe poder realizar a dos niveles distintos:

1- Nivel red.

Se debe poseer un sistema capaz de realizar búsquedas de objetos de aprendizaje distribuidos en diversos repositorios de una red. Para realizar dicha tarea es necesario disponer de los metadatos que caracterizan a los objetos buscados y de las direcciones de registros de contenidos donde localizar los repositorios donde se encuentran los contenidos buscados. La implementación del sistema puede plantearse de dos formas distintas:

a. *Búsqueda síncrona*

El programa de búsqueda visita cada uno de los registros de contenidos buscando los contenidos que concuerdan con los parámetros de la búsqueda.

b. *Búsqueda asíncrona.*

En esta búsqueda se consideran dos roles o tipos de entidades, por una parte las entidades que buscan unos determinados objetos, y otras entidades que ofrecen unos determinados objetos. Las primeras entidades tienen la obligación de publicar(es decir hacer accesibles) las características o metadatos de los objetos de aprendizaje que están buscando, y a su vez las segundas entidades tienen la obligación de publicar los metadatos de los objetos de aprendizaje ofertados. Con esta información un sistema de notificación de eventos se encargara de transmitir esta información entre los diversos registros de contenidos, de forma que en cada registro de contenidos se debe analizar el grado de similitud que existe entre lo ofertado y lo demandado. Cuando el grado de similitud supere un cierto umbral entonces se realizará una notificación al demandante para informarle que se ha encontrado contenido con las características buscadas, con lo cual se podrá conectar al repositorio de contenidos encontrado y realizar la recuperación de los contenidos si procede.

Implementar un sistema de búsqueda de tales características implica resolver varias cuestiones de diseño:

1) Topología de la red.
Es necesario analizar si la configuración en que las máquinas que contienen a los distintos registros de contenidos afecta o no a la búsqueda, y en caso de afectarla ver que configuración es la más interesante.

2) Algoritmo de enrutamiento.
Hay que seleccionar de toda la información que se dispone, cuál será necesaria transmitir para pode llevar a cabo las búsquedas de una forma correcta y eficiente.

3) Estrategia de procesamiento.
Se deben estudiar si existe alguna heurística que permita optimizar las búsquedas sobre la red.

2- Nivel de repositorio de contenidos.

Localizado un repositorio de contenidos, una de las interfaces que un repositorio debería ofrecer para su uso sería un mecanismo de búsqueda y recuperación. La búsqueda debe realizarse a partir de los metadatos o información característica de los contenidos solicitados, y podrá ser una búsqueda flexible o no, en el sentido de buscar objetos que tengan exactamente todas las características requeridas o bien objetos cuyo grado de similitud con los buscados supere un cierto umbral de adecuación.

La búsqueda a nivel local en repositorios plantea otra cuestión a resolver por el hecho de que un objeto de aprendizaje pueda estar formado a su vez por otros objetos de aprendizaje más simples. En este sentido los metadatos de los objetos de aprendizaje quedarían ocultos por el objeto de aprendizaje mayor que los contiene. Es por ello que también habría que definir que tipo de búsqueda es la que se quiere realizar:

- Búsqueda superficial.
  Dicha búsqueda se realiza únicamente sobre los metadatos de los objetos de aprendizaje compuestos, y almacenados como tal en el repositorio.

- Búsqueda interna.
  Dicha búsqueda se realiza sobre cada objeto de aprendizaje compuesto, y sobre todos los objetos de aprendizaje que lo componen hasta llegar al objeto más simple posible (la unidad mínima de contenido que se considere ya un objeto de aprendizaje y como tal tenga ya asociados unos metadatos). Este tipo de búsqueda conllevará realizar un desempaquetado de los objetos de contenido.

7.2.3 Etiquetado automático de los recursos

Una labor en la creación de un objeto de aprendizaje es la definición del conjunto de metadatos que caracterizan al objeto o también llamado etiquetado del objeto. El que un objeto este bien etiquetado significará el que se puedan realizar búsquedas sobre los objetos en base al cumplimiento o posesión de ciertos valores para unos metadatos dados o el que sea factible la reutilización de los objetos para contextos diferentes que el que fueron creados inicialmente. Por tanto el etiquetado juega un papel esencial en los objetos de aprendizaje, sin embargo la realización del mismo es una tarea complicada, delicada y que consume mucho tiempo si se quiere realizar correctamente. Es por ello por lo que se plantea la posibilidad de poder llevar a cabo una clasificación o etiquetado con cierto grado de automatización.

El planteamiento es llevar a cabo un etiquetado parcial que posteriormente sea supervisado por un usuario humano, el cual dé el visto bueno o modifique los valores asignados, y añada valores a los metadatos no etiquetados. La forma de implementar este etiquetado parcial se basaría en llevar a cabo una clasificación previa de los metadatos en:

- ✓ Metadatos automatizables directamente.
  Aquellos cuyo valor puede obtenerse directamente del objeto o recurso.

- ✓ Metadatos automatizables indirectamente.
  Aquellos cuyo valor se obtiene a partir del procesamiento de la información que puede obtenerse directamente del objeto o recurso.

- ✓ Metadatos no automatizables.
  Aquellos cuyo valor sólo puede ser obtenido mediante la intervención directa del usuario humano.

Para llevarla a cabo será necesario analizar qué información puede obtenerse de una forma sencilla y directa del tipo de objetos de aprendizaje o recursos con los que se va a trabajar (según el tipo de objeto o recurso se podrá conocer un tipo y cantidad diferente de información). Es por ello que la clasificación estará en dependencia del tipo de objetos de aprendizaje o recursos con que se vayan a tratar. Conocida dicha información se podrá seleccionar que metadatos pueden ser etiquetados directa y automáticamente a partir del conocimiento de la misma.

También habrá un conjunto de metadatos que aún no pudiendo ser rellenados directamente, si pueda automatizarse su definición. Los valores de estos metadatos se deducirán a partir de la aplicación de un sistema "inteligente" que procese la información extraída a los objetos o recursos. Un sistema de tales características podría basarse en técnicas de razonamiento basado en casos (CBR), de forma que si se posee una base de información con los etiquetados que se hayan hecho en situaciones parecidas, pueda compararse el caso de etiquetado que se vaya a hacer con estos etiquetados anteriores y según se llegue o no a un grado de similitud definido por cierta función de comparación, entonces se etiquete de una forma u otra el objeto. También podría pensarse en técnicas basadas en reglas que procesarán la información, y de acuerdo al cumplimiento de una regla u otra para dicho metadato, se le asigna un valor u otro al metadato.

Por último habrá un conjunto de metadatos cuya asignación de valores no sea posible sin la intervención previa del usuario humano que juzgue cual es el valor que se le debe asignar.

7.2.4 Mejoras en las funcionalidades de edición de la herramienta

Hay cuatro mejoras inmediatas sobre la forma en que se realiza la edición en la herramienta:

*1.- Edición de Metadata***.**

Actualmente la edición de los metadatos correspondientes a la especificación de Metadata, se realizan directamente sobre el árbol principal asociado al objeto de aprendizaje. Debido a la extensión de estos metadatos (disponen de una especificación propia aparte) sería razonable que su edición se realizará en un formulario aparte en el que fuera más fácil para el usuario modificar sus datos, y pudiera ver de un solo vistazo los contenidos almacenados en estos metadatos.

*2.- Multiedición de Objetos de Aprendizaje***.**

En estos momentos sólo es posible realizar la edición de un solo objeto de aprendizaje a la vez. Una mejora substancial sería la posibilidad de poder abrir y editar simultáneamente varios objetos de aprendizaje a la vez.

*3.- Personalización de la presentación de los contenidos***.**

Tanto la forma de estructurar los contenidos de un objeto de aprendizaje como su forma de mostrarlos es dependiente de una hoja de estilos que es independiente de la aplicación. Cabría mejorar este punto permitiendo al usuario que pudiera elegir la hoja de estilo que quisiera usar para la presentación de los contenidos.

*4.- Adaptación Web de la herramienta***.**

El futuro de este tipo de herramientas pasa por su adaptación a tecnologías de tipo Web, de forma que sean accesibles de forma universal a cualquiera con solo disponer de un navegador, y que puedan de esta forma interaccionar con repositorios de contenidos conectados a la red. De esta forma se permitirá una reutilización de objetos ya realizados, así como una rápida generación de otros nuevos.

APÉNDICE A

UML

A.1 Diagramas de clases

A.1.1 Módulo 1: Herramienta de autoría

A.1.1.1 Diagrama de Clases Herramienta de Autoría

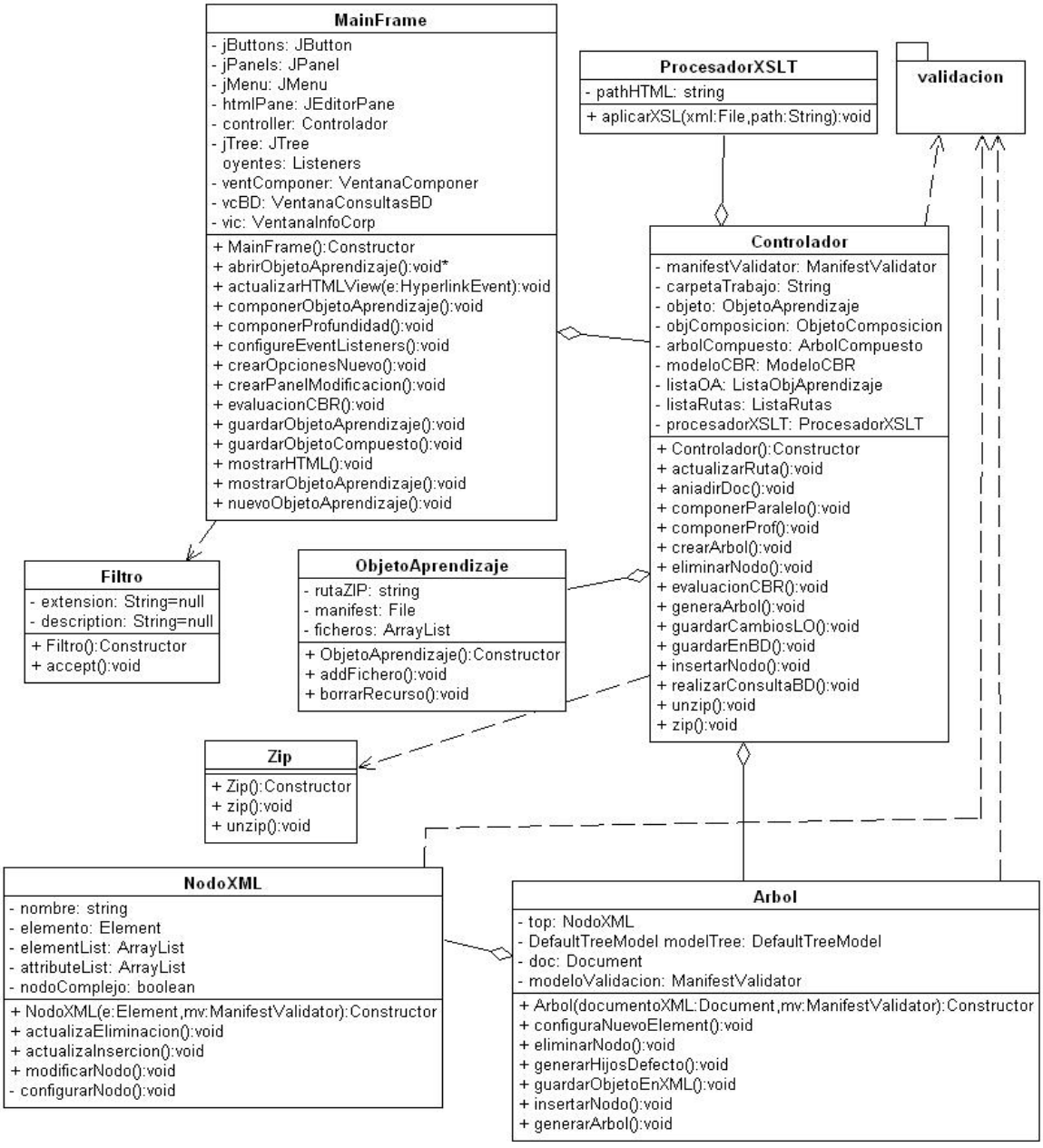


Diagrama A.1.1.1.- Diagrama de Clases Herramienta de Autoría

Este diagrama de clases corresponde a la estructura principal de la herramienta de autoría. Se trata de la organización y relación jerárquica de clases del modelo de la aplicación, donde guardaremos la representación del objeto de aprendizaje que estemos tratando.

Como puede observarse, el diseño permite diferenciar claramente lo que llamaríamos la interfaz de usuario de la parte más interna del modelo. Como intermediario entre ambas partes figura nuestra clase Controlador. La interfaz de usuario no posee información relevante por si misma, recurriendo al Controlador cada vez que el usuario a través de dicha interfaz realiza alguna operación. Todas las peticiones que la interfaz, a instancia del usuario, realiza, son gestionadas por el Controlador, el cual accede al modelo para llevar a cabo dicha petición. Con esto conseguimos que cualquier modificación en la interfaz a nivel de estilo no afecte al modelo, que permanece independiente. Así mismo, cualquier modificación en la implementación de nuestro modelo será transparente a la interfaz, que solo debe negociar con el Controlador.

Esto es a nivel general el diseño Modelo-Vista-Controlador del que nos valemos para desarrollar la herramienta LOMEditor.

Entrando más en profundidad en los detalles de las clases vemos que nuestro objeto de aprendizaje será almacenado de la siguiente forma: Por un lado, los datos generales del objeto, es decir, a nivel físico, como puedan ser el nombre, la ruta, el fichero del manifiesto, y una lista de recursos, etc., serán almacenados en la clase ObjetoAprendizaje, mientras que por otro lado, la representación más teórica o abstracta del objeto de aprendizaje, será guardada en forma de árbol en la clase Arbol. Este árbol, como se explica en los detalles de implementación, permite aprovechar las facilidades del Modelo Delegado de Swing, ya que implementa la interfaz de árbol de Java, logrando con ello poder construir el árbol JTree de la interfaz. A partir de entonces, cualquier actuación sobre el JTree de la interfaz será aplicado a su modelo interno.

Para la construcción de esta clase Arbol son necesarios unos nodos especiales, del tipo NodoXML, que guardarán toda la información del Elemento del manifiesto al que representan. Además, gracias a la independencia de esta herramienta respecto a la gramática de partida, almacenará meta-información acerca de dichos elementos. Esta información, básica para la correcta edición del objeto de aprendizaje, será extraída del paquete de validación, construido inicialmente a partir de la gramática del objeto de aprendizaje en tratamiento.

Finalmente podemos encontrar otras clases de menor importancia que serán necesarias para ciertas operaciones puntuales, como pueden ser la clase Filtro, útil en la interacción con el usuario mediante cuadros de diálogo, o la clase Zip, que nos ayudará con la gestión de ficheros comprimidos, ya sea al abrirlos o guardarlos.

A.1.1.2 Diagrama de Clases Paquete de Validación

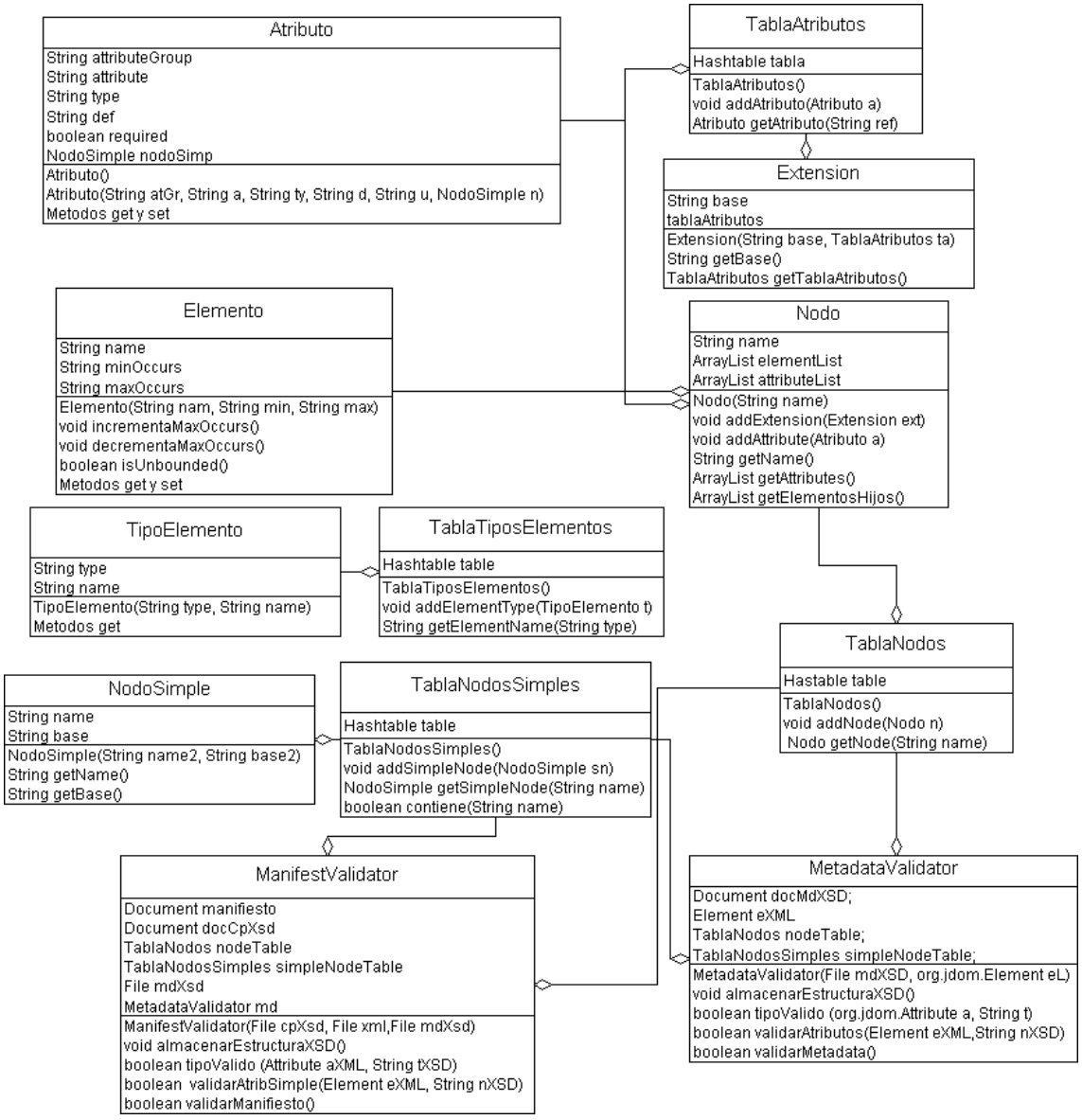


Diagrama A.1.1.2.- Diagrama de Clases Paquete de Validación

Este importante paquete de clases Java será el encargado de otorgar a la herramienta LOMEditor la capacidad de adaptarse a las gramáticas que cada objeto de aprendizaje haya seguido en su construcción. Esta gramática deberá ser aportada junto al objeto de aprendizaje basado en la misma, en el fichero .zip, como documento "xsd".
(ficheros xsd correspondientes a la versión del IMS Content Packaging e IMS Metadata sobre la que se basa el objeto de aprendizaje)

La primera acción que realiza LOMEditor ante la presencia de un objeto de aprendizaje es validar su manifiesto, de acuerdo con la gramática en la que se basa. Para ello debe primero construir un modelo de validación que represente dicha gramática, para más tarde analizar el manifiesto.

Cada etiqueta de la gramática reflejada en la xsd es procesada y analizada por un determinado grupo de clases; lo vemos a continuación:

- xsd:element: Esta etiqueta será procesada por las clases TipoElemento y TablaTiposElementos, que guardarán, entre otra información, el nombre y tipo de cada elemento encontrado. Todos estos elementos serán guardados en una tabla para su posterior acceso y utilización.

- xsd:attributeGroup: Para esta otra etiqueta, serán principalmente las clases Atributo y TablaAtributos las encargadas del procesamiento. En este caso se trata de analizar y guardar los diferentes atributos encontrados en la gramática, así como los detalles de los mismos.

- xsd:complexType: Las clases Elemento, Nodo y TablaNodos serán esta vez las que procesen las etiquetas que guardan la información de los elementos complejos. Será aquí donde encontremos la meta-información necesaria en la edición del objeto de aprendizaje, como los posibles hijos o atributos de cada elemento.

- xsd:simpleType: Esta última etiqueta será procesada por las clases NodoSimple y TablaNodosSimples, que guardarán, análogamente a lo anterior, la información de los elementos de tipo simple del objeto de aprendizaje.

Para poder utilizar toda esta importante información, recurriremos a la clase ManifestValidator, que almacenará dichos datos. Guardará además, por separado, la información de la gramática a utilizar para los metadatos del objeto de aprendizaje.

Una vez construido este paquete de validación podremos no solo comprobar la corrección del objeto de aprendizaje introducido, sino que también será posible crear, eliminar o modificar elementos en el mismo.

A.1.2 Módulo 2: Módulo de composición

A.1.2.1 Diagrama de Clases Paquete de Composición

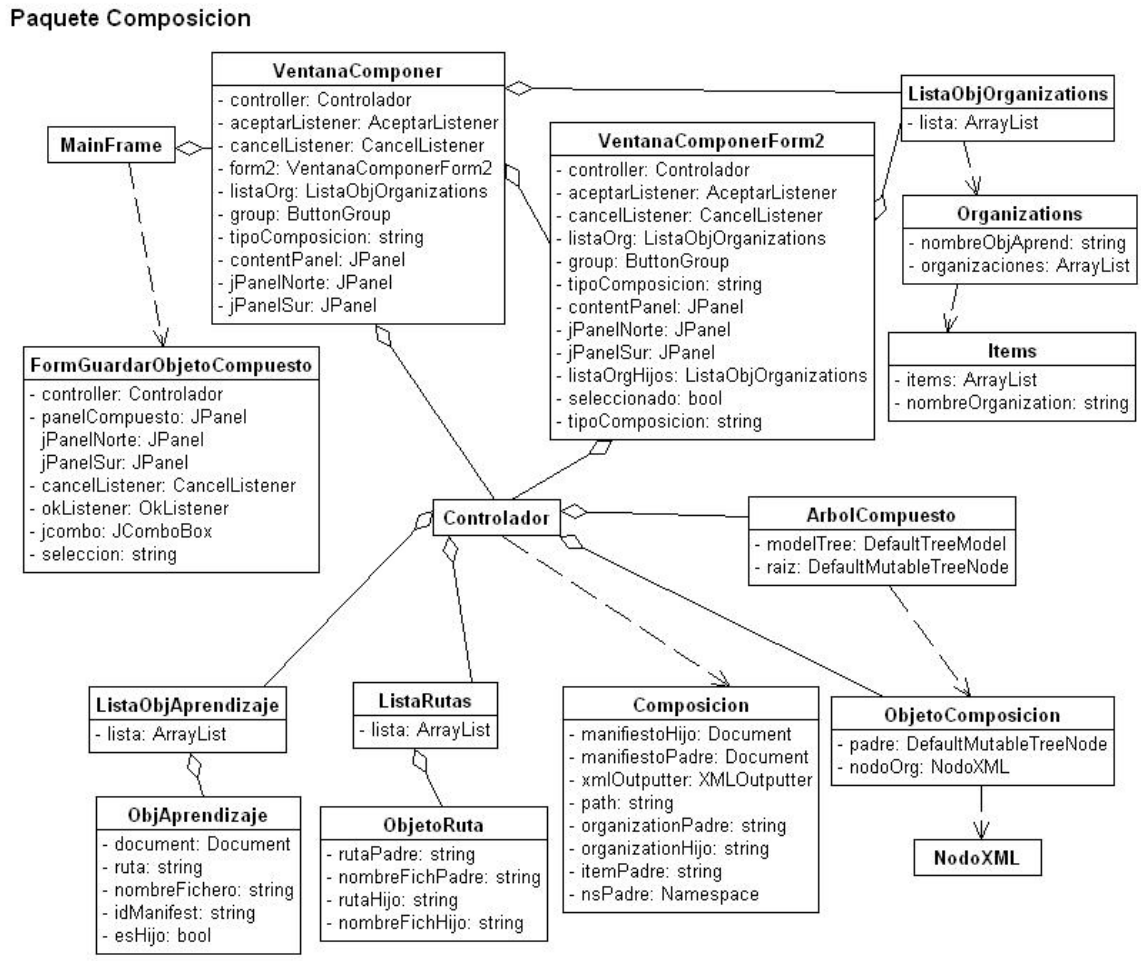


Diagrama A.1.2.1.- Diagrama de Clases Paquete de Composición

Este diagrama de clases corresponde al módulo de composición de la aplicación. Se encarga de implementar la tarea de composición de objetos de aprendizaje.

Dicho módulo, al igual que el resto de la arquitectura de la aplicación, respeta el patrón de Ingeniería del Software conocido como Model-View-Controller. De esta forma, el módulo está concebido de manera que se pueda dividir en tres partes fundamentales: Modelo, Vista y Controlador.

Las clases que forman el modelo son las siguientes:

- Composicion: Se encarga de la composición de los objetos que le llegan desde el Controlador y que fueron seleccionados por el usuario en las ventanas de la interfaz.

- ArbolCompuesto: Representa el árbol en el que se muestran los objetos de composición que se van creando al abrir el usuario los objetos de aprendizaje (con la opción Mostrar objeto compuesto). Contiene dos atributos: DefaultTreeModel (representa el modelo del árbol) y DefaultMutableTreeNode (es el nodo raíz del árbol, del que colgarán todos los objetos que se vayan abriendo).

- ObjetoComposicion: Este objeto es creado a partir del objeto Arbol (creado a su vez a partir del objeto de aprendizaje que quiere mostrar el usuario para su posible composición). Sus atributos son: DefaultMutableTreeNode (nodo raíz de este árbol y del que cuelga el NodoXML) y NodoXML (es el nodo Organizations del Arbol de este objeto de aprendizaje).

- ListaRutas: Lista de objetos de tipo ObjetoRuta.

- ListaObjAprendizaje: Lista de objetos de tipo ObjAprendizaje.

- ObjetoRuta: Contiene los siguientes atributos para realizar la composición: el nombre de los ficheros padre e hijo y las rutas respectivas.

- ObjAprendizaje: Clase en la que se almacenan datos correspondientes al objeto de aprendizaje abierto por el usuario. Los atributos que contiene son:
  document del manifiesto del objeto de aprendizaje actual,
  ruta original del objeto de aprendizaje,
  idManifest del manifiesto de dicho objeto, y
  nombreFichero de la carpeta que lo contiene.

El Controlador, será el mismo que se utiliza en el resto de la aplicación, lo que facilitará la unión de la aplicación como un ente único.

La parte correspondiente a la vista de este módulo, está compuesta por las siguientes clases:

- VentanaComponer: Esta clase interactúa con el usuario mostrándole una ventana con todos los objetos (y sus organizaciones, en el caso de la composición en paralelo; o los items de cada organización en el caso de la composición en profundidad) abiertos hasta el momento y que están mostrados en el árbol de composición del panel izquierdo de la aplicación. Las organizaciones de cada objeto en la composición en paralelo, y los items

de cada organización en la composición en profundidad, van acompañadas por una pestaña, de las cuales sólo se podrá elegir una. El usuario deberá seleccionar una organización y aceptar en esta ventana.
La organización o item elegido será el padre (y el objeto correspondiente será el objeto contenedor) en la futura composición.
La aceptación de esta ventana provocará el cierre de la misma y la creación de un objeto VentanaComponerForm2 que será explicado a continuación.

- VentanaComponerForm2: Con esta clase se interactúa con usuario mostrándole las organizaciones de todos los objetos mostrados (tanto para la composición en paralelo como para la composición en profundidad) excepto el objeto elegido como padre en la ventana anterior (VentanaComponer). El procedimiento es el mismo: el usuario selecciona una organización que será tomada como hijo para la composición (y, por tanto, el objeto al que pertenezca será el objeto contenido). Acepta en la ventana y esto provocará la composición de ambos objetos.

- FormGuardarObjetoCompuesto: Esta clase interactúa con el usuario mostrándole una ventana con todos los objetos compuestos hasta el momento. El usuario elige uno de ellos y la ruta destino en la que quiere guardar dicho objeto comprimido. Una vez aceptado, dicho objeto es guardado en formato comprimido (zip) en la ruta seleccionada por el usuario, y en caso de que se hubiera compuesto más de un objeto, se pregunta al usuario si desea guardar otro objeto.
Si la respuesta es afirmativa, se vuelve a mostrar esta ventana, sino se cierra.

- ListaObjOrganizations: Lista de objetos de tipo Organizations.

- Organizations: Objeto que contiene una lista en la que se guardan todos los JCheckBox que son mostrados en VentanaComponer. Gracias a esta lista conoceremos el objeto que ha seleccionado el usuario y mostraremos los mismos JCheckBox (excepto el ya elegido como padre) en VentanaComponerForm2.

- Items: En el caso de la composición en profundidad, la lista de la clase Organizations contiene objetos de este tipo.

A.1.3 Módulo 3: Módulo de evaluación de calidad

A.1.3.1 Diagrama de Clases Paquete CBR

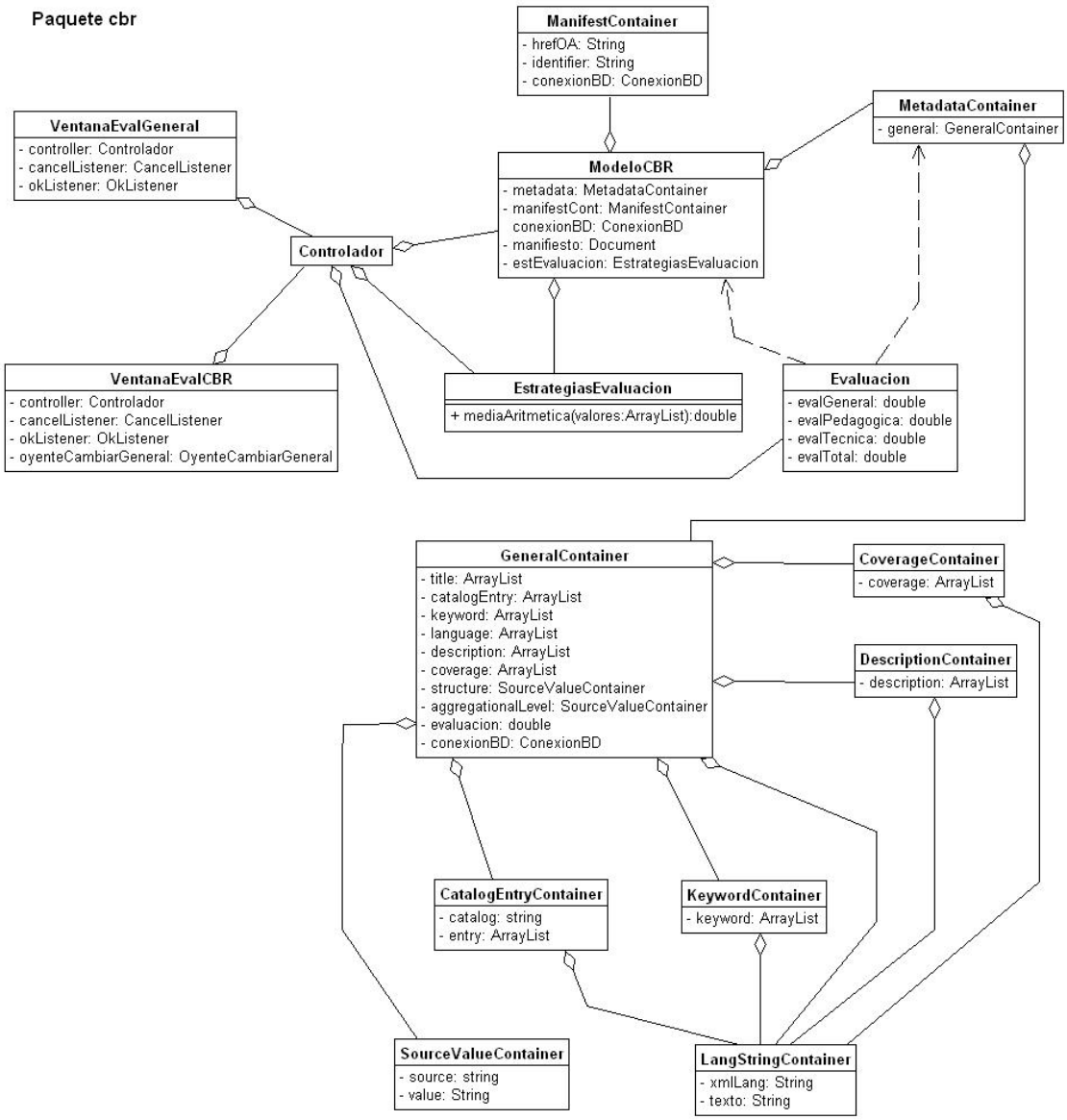


Diagrama A.1.3.1.- Diagrama de Clases Paquete CBR

Este módulo de la aplicación se encarga de implementar la tarea de evolución de calidad de los objetos de aprendizaje, utilizando la técnica de Inteligencia Artificial conocida como CBR.

El módulo, al igual que el resto de la aplicación, respeta el patrón de Ingeniería del Software conocido como Model-View-Controller. De esta forma, el módulo está concebido de manera que se pueda dividir en tres partes fundamentales: Modelo, Vista y Controlador.

Con este objetivo, la clase ModeloCBR se encarga de almacenar toda la información necesaria para realizar este tipo de evaluaciones, tanto de manera manual como de manera automática. Esta clase contiene cinco atributos básicos en la evaluación CBR:

- manifiesto::Objeto de la clase Document: almacenará la estructura del manifiesto asociado al objeto de aprendizaje, que se mapeará a clases java para facilitar su procesamiento. Las clases java que contendrán esta estructura son las que hemos denominado clases *Container,* siguiendo la signatura: *Nombre del elemento que mapean + Container*.

- metadata::Objeto de la clase MetadataContainer: que contendrá toda la información relevante a los metadatos del objeto de aprendizaje. Esta información será mapeada automáticamente del manifiesto del objeto, a través del objeto Document anteriormente comentado.

- manifestCont::Objeto de la clase ManifestContainer: Este objeto almacenará contenido relevante a la etiqueta raíz del manifiesto, manifest. Además, contendrá la ruta en la que se ubica el objeto de aprendizaje, para hacer posible el nexo de unión entre la funcionalidad de búsquedas en la base de datos de objetos de aprendizaje y la herramienta de autoría.

- conexionBD::Objeto de la clase ConexionBD: Se encargará de ofrecer la interfaz de acceso a la base de datos del sistema, a través de la cual se gestionarán las consultas de selección, inserción y modificación sobre la misma.

- estEvaluacion::Objeto de la clase EstrategiasEvaluación: Contiene todas las métricas utilizables a la hora de evaluar la calidad de los objetos de aprendizaje. Se implementará como una clase independiente para facilitar la ampliación y mejora de éstas técnicas en un futuro.

El Controlador, será el mismo que se utiliza en el resto de la aplicación, lo que facilitará la unión de la aplicación como un ente único.

La parte correspondiente a la interfaz de este módulo, está compuesta por dos ventanas de interacción con el usuario que se conectan al Modelo a través del Controlador cómo indican las directrices del Model-View-Controller.

- VentanaEvalGeneral: Clase que está directamente relacionada con la evaluación manual de objetos de aprendizaje. Se genera automáticamente a través del mapeado del manifiesto del objeto, y muestra al usuario las distintas etiquetas evaluables y los posibles valores en la evaluación, para que este los rellene a su antojo.

- VentanaEvalCBR: Esta clase es la encargada de mostrar al usuario una interfaz en la que se indique el resultado de la evaluación automática, y ofrecerle además la posibilidad de cambiar esta evaluación, accediendo a la ventana comentada anteriormente (VentanaEvalGeneral) o aceptar la inferida por el sistema mediante la técnica de CBR. Está aceptación provocará el aprendizaje automático del sistema.

A.2 Diagramas de secuencia

A.2.1 Módulo 1: Herramienta de autoría

A.2.1.1 Secuencia "Nuevo objeto de aprendizaje"

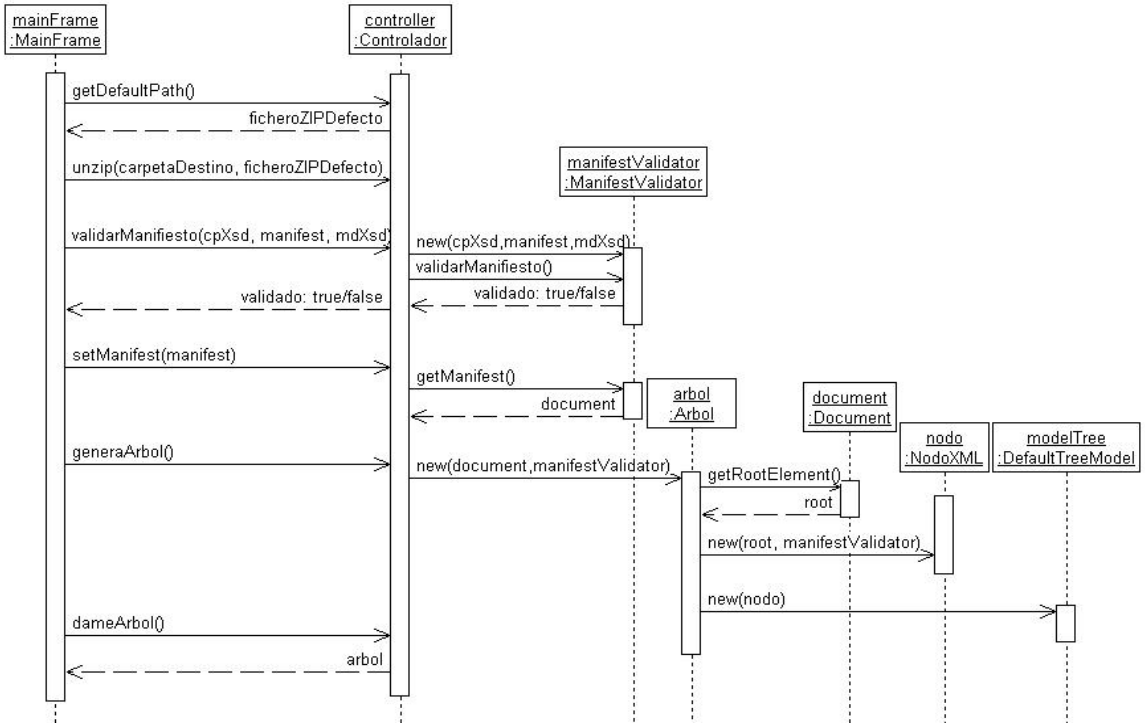


Diagrama A.2.1.1 Secuencia "Nuevo objeto de aprendizaje"

Este diagrama muestra la secuencia de pasos que sigue la herramienta LOMEditor para crear un nuevo objeto de aprendizaje.

El usuario realizaría la petición sobre la interfaz de nuevo objeto de aprendizaje. La interfaz le pide el destino del objeto, es decir, donde será descomprimido y tratado. Con esta información, la interfaz manda al controlador que descomprima el objeto mínimo por defecto destinado a este fin, en la carpeta destino que aportó el usuario.

A partir de aquí, el proceso a seguir es el siguiente: Se valida el objeto de aprendizaje, en este caso más que para comprobar su corrección, únicamente para obtener los datos de su gramática.

Generamos más tarde el árbol que representará al objeto, a partir del manifiesto mínimo del objeto por defecto y la estructura del paquete de validación, referente a la gramática en la que estará basado. Finalmente, se muestra al usuario el JTree del objeto para comenzar a construirlo.

A.2.1.2 Secuencia "Abrir objeto de aprendizaje"

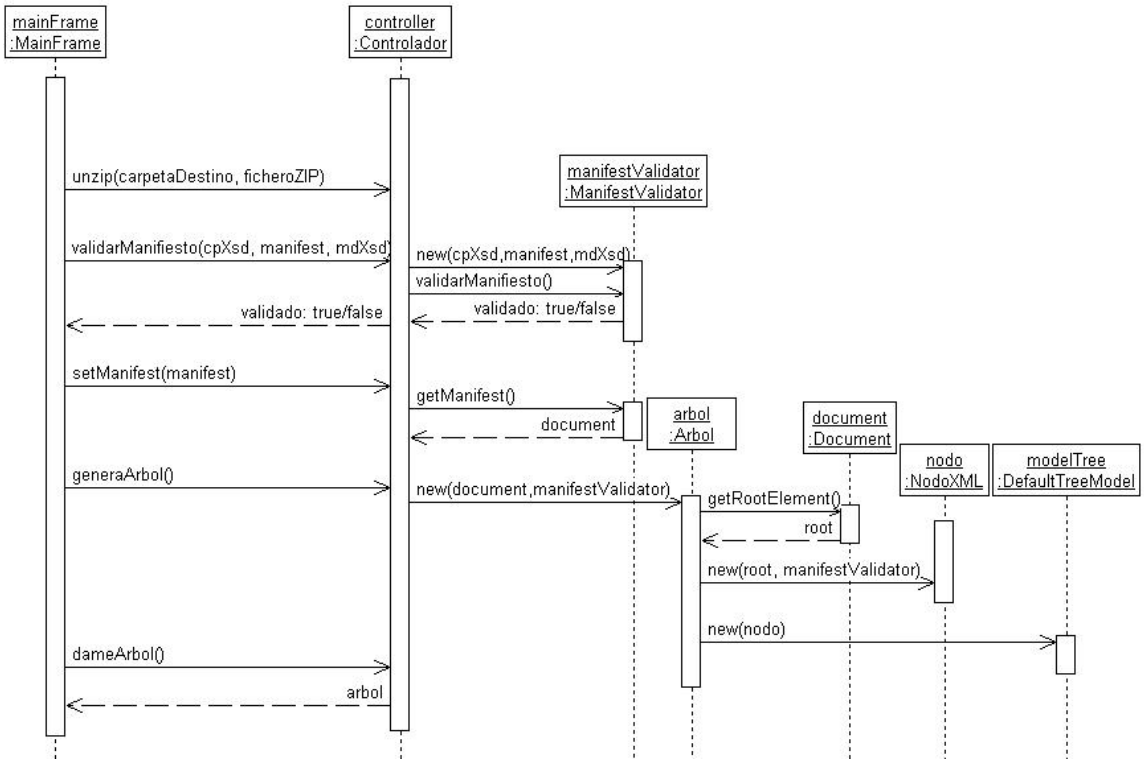


Diagrama A.2.1.2 Secuencia "Abrir objeto de aprendizaje"

Diagrama que muestra la secuencia de pasos que sigue la herramienta LOMEditor para abrir un objeto de aprendizaje.

Muy similar al nuevo objeto de aprendizaje, con la salvedad de que ante la petición "Abrir" del usuario se le ha de pedir tanto el objeto (en formato ZIP) como la carpeta destino de trabajo.

A partir de aquí, se seguirán los mismos pasos que seguimos en el caso de "nuevo objeto de aprendizaje". Hay que tener en cuenta que esta vez la validación cobra mayor importancia, ya que no sabemos de antemano que el objeto está correctamente construido.

Finalmente, se muestra al usuario el árbol JTree que representa el objeto de aprendizaje abierto, permitiendo su edición.

A.2.1.3 Secuencia "Guardar objeto de aprendizaje"

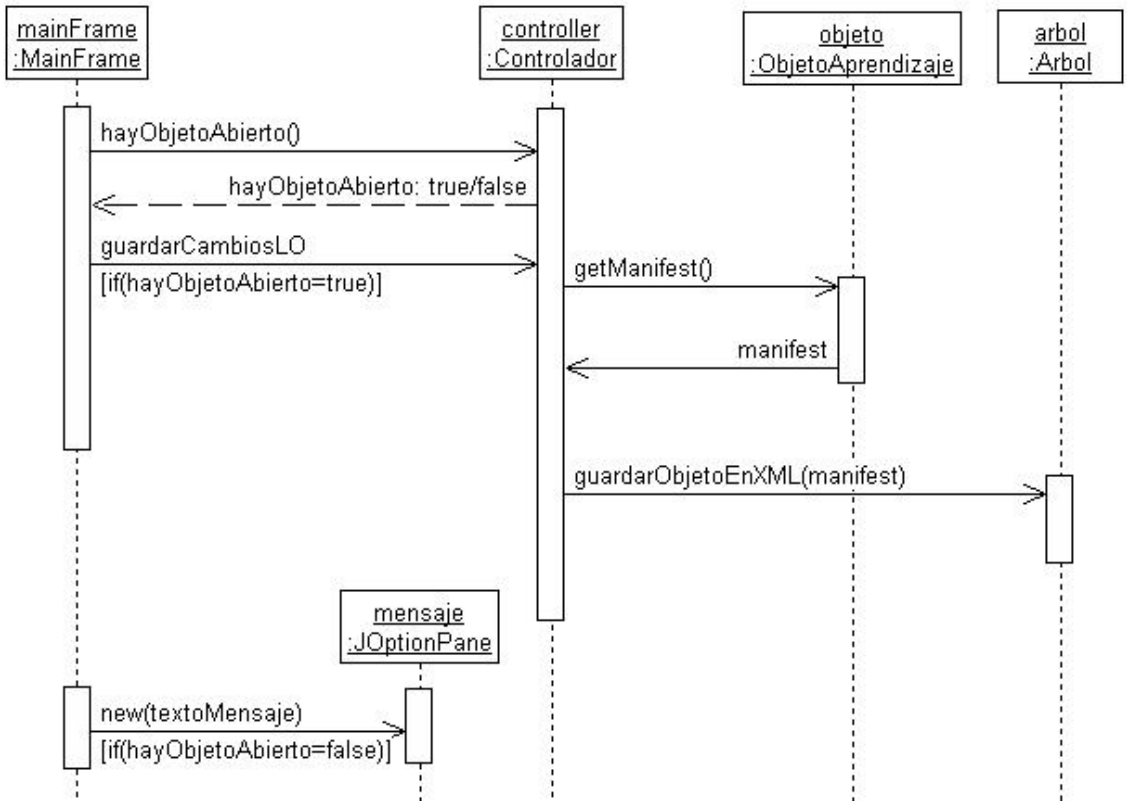


Diagrama A.2.1.3 Secuencia "Guardar objeto de aprendizaje"

Diagrama que muestra la secuencia de pasos que sigue la herramienta LOMEditor para guardar un objeto de aprendizaje abierto.

El usuario indicará a LOMEditor que desea guardar el objeto de aprendizaje que se encuentra actualmente en edición. La interfaz hará esa misma petición al controlador, el cual, al poseer toda la información relativa al objeto, tanto a nivel físico en su atributo "objeto", como a nivel lógico en el "árbol", ejecutará la operación.

Para este guardado es necesaria la intervención de JDom, que permite escribir un documento XML (guardado en el árbol) en un fichero físico (cuya ruta conoce "objeto").

Finalmente el objeto quedará salvado, habiendo guardado su manifiesto.

A.2.1.4 Secuencia "Insertar nuevo Elemento en el objeto de aprendizaje"

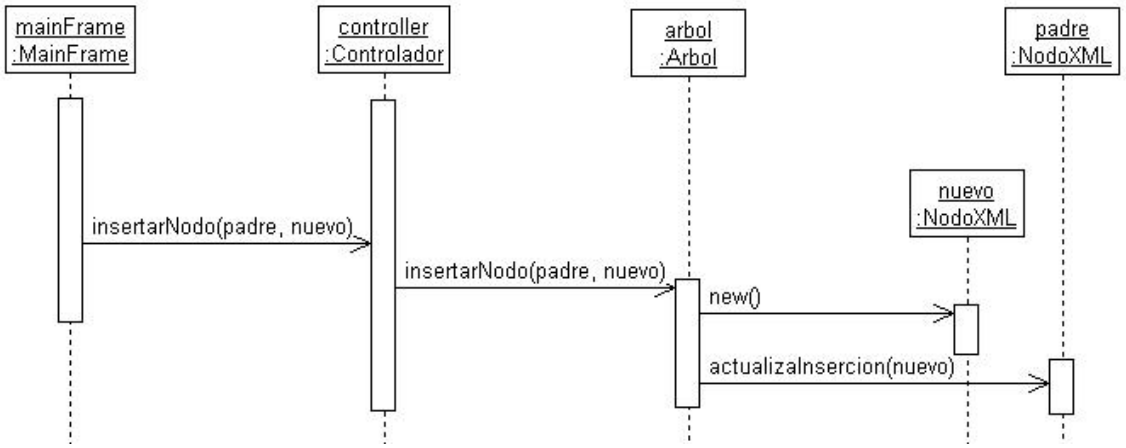


Diagrama A.2.1.4 Secuencia "Insertar nuevo elemento en el objeto de aprendizaje"

Diagrama que muestra la secuencia de pasos que sigue la herramienta LOMEditor para insertar un nuevo elemento en un objeto de aprendizaje abierto.

El usuario debe indicar, mediante la interfaz, que nuevo nodo desea añadir al objeto, escogiendo por tanto, al padre del mismo.

La interfaz enviará la petición al controlador, con los datos anteriormente obtenidos, para que este último realice las operaciones pertinentes sobre el modelo.

Finalmente lograremos contar con un elemento más en el objeto de aprendizaje, reflejándose esto sobre el árbol que visualiza el usuario en la interfaz.

A.2.1.5 Secuencia "Eliminar elemento del objeto de aprendizaje"

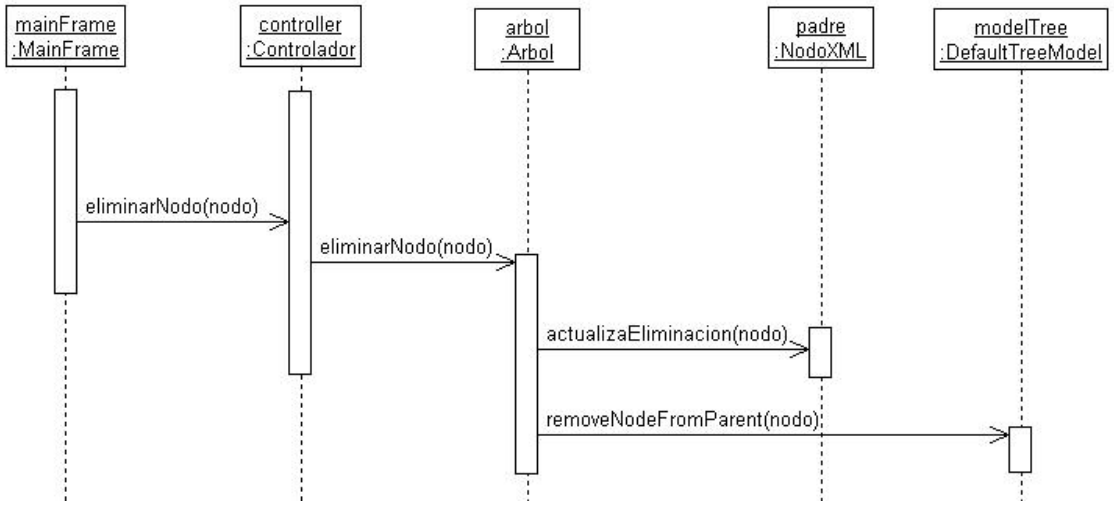


Diagrama A.2.1.5 Secuencia "Eliminar elemento del objeto de aprendizaje"

Diagrama que muestra la secuencia de pasos que sigue la herramienta LOMEditor para eliminar un elemento de un objeto de aprendizaje abierto.

El usuario debe indicar, mediante la interfaz, que nodo desea eliminar del objeto de aprendizaje.

La interfaz enviará la petición al controlador, con los datos anteriormente obtenidos, para que este último realice las operaciones pertinentes sobre el modelo.

Finalmente habremos conseguido eliminar un elemento del objeto de aprendizaje, hecho que se verá reflejado en el árbol de la interfaz de usuario.

A.2.2 Módulo 2: Módulo de composición

A.2.2.1 Secuencia "Mostrar objeto de aprendizaje"

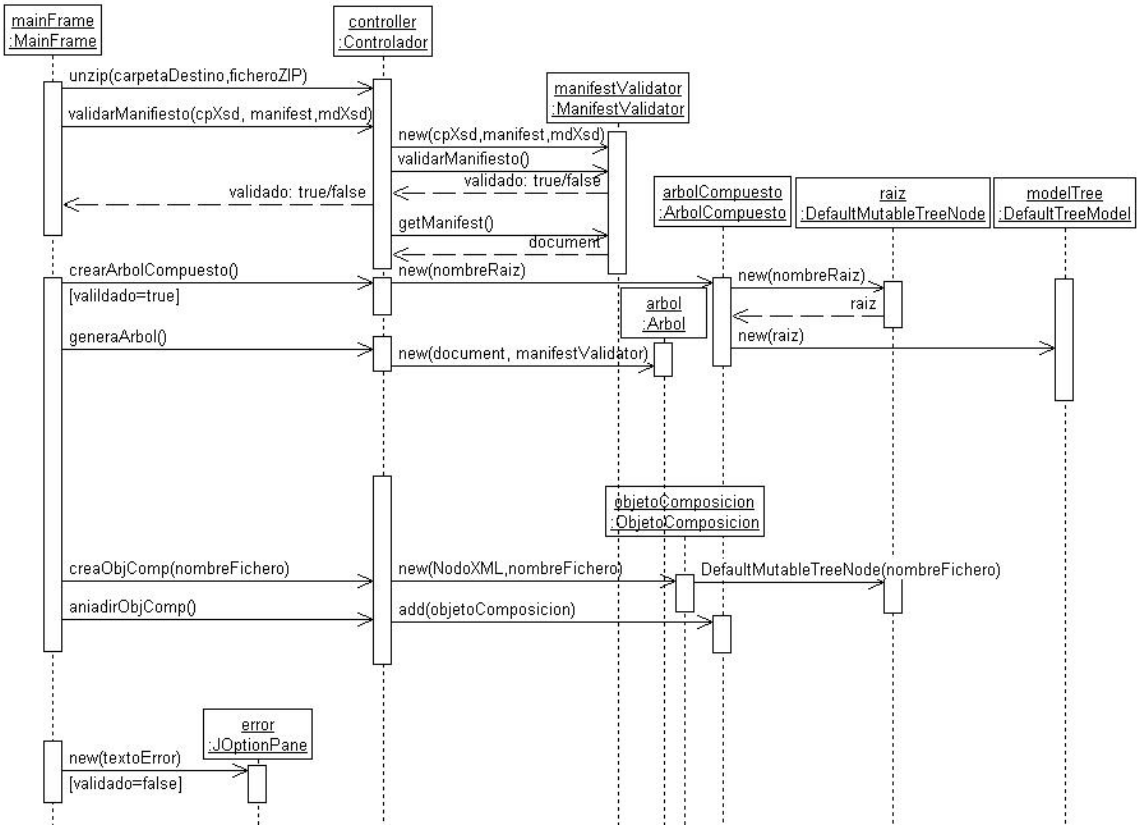


Diagrama A.2.2.1 Secuencia "Mostrar objeto de aprendizaje"

En este diagrama se muestra la secuencia de pasos que sigue la herramienta LOMEditor para mostrar un objeto de aprendizaje.

Tras realizar el usuario la petición sobre la interfaz de mostrar un objeto de aprendizaje, se le pedirá el fichero comprimido en formato zip (*ficheroZIP* en el diagrama de secuencia), para poder realizar la descompresión del mismo y, posteriormente, su validación.

Al descomprimir dicho objeto se crea una carpeta temporal con el nombre del mismo y sobre la que se trabajará posteriormente durante la composición. Si ya existe una carpeta temporal con ese nombre (porque ya se haya abierto un objeto con el mismo nombre, aunque no necesariamente el objeto coincidirá) se pregunta al

usuario si desea continuar con la apertura del objeto, puesto que se sobrescribirán los datos de dicha carpeta.

La validación es un paso importante puesto que no sabemos de antemano si el objeto está correctamente construido o si realmente el usuario está intentando mostrar un objeto de aprendizaje. Además de que de esta forma obtenemos los datos de su gramática, necesarios para los pasos siguientes.

Si el objeto no fuera validado, devolvería *false,* y tal como se indica en el diagrama, se mostraría un mensaje de error al usuario advirtiéndole que no se puede mostrar el objeto seleccionado puesto que no ha sido validado.

En caso contrario (que el objeto fuera validado), se comprueba que dicho objeto no está mostrado ya en el árbol de composición. Si lo estuviera se muestra un mensaje de error informando al usuario que el objeto ya está mostrado en el árbol de composición. Sino, se crea el árbol de composición (ArbolCompuesto) en el que se mostrarán los objetos de composición.

Se genera el árbol (de la clase Arbol) a partir del document y del manifestValidator obtenidos en la validación del objeto de aprendizaje.

Se crea un objeto de composición (ObjetoComposicion) a partir del NodoXML obtenido del árbol (que será el nodo Organizations del objeto de aprendizaje tratado).

Por último, se añade este objeto de composición al árbol de composición, y se actualiza el JTree, mostrando al usuario el objeto de aprendizaje como un objeto de composición.

A.2.2.2 Secuencia "Composición en paralelo"

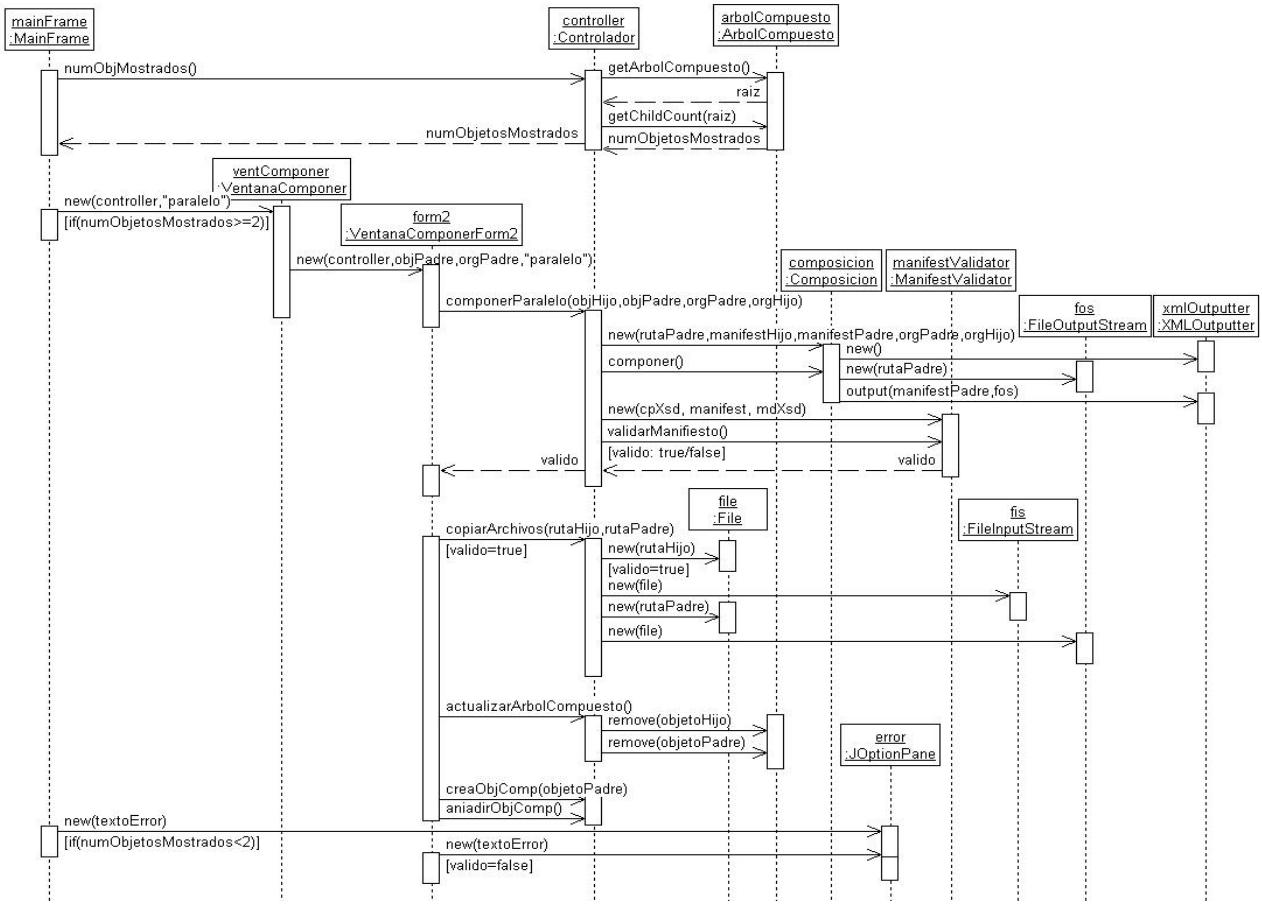


Diagrama A.2.2.2 Secuencia "Composición en paralelo"

Este diagrama muestra la secuencia de pasos que sigue la herramienta LOMEditor para realizar la composición en paralelo.

El usuario pulsa en la opción de composición en paralelo de dos objetos de aprendizaje.

Se comprueba si previamente el usuario ha abierto al menos dos objetos de aprendizaje (mostrados ahora en el árbol de composición como objetos de composición). Si esto es así se procederá a pedir al usuario los datos necesarios para la composición, sino, se mostrará un mensaje de error advirtiendo que no se puede realizar composición debido a que para ello son necesarios al menos dos objetos de composición.

A continuación explicamos los pasos seguidos al realizar la composición:

Primeramente se muestra una ventana (*ventComponer* en el diagrama de secuencia) en la que el usuario deberá elegir la organización (y por tanto el objeto) que actuará como objeto contenedor o padre.

Al aceptar en esta ventana aparecerá otra en la que el usuario deberá elegir la organización (y consecuentemente el objeto) que actuará como objeto contenido en la composición.

Una vez aceptada esta otra ventana por el usuario, se componen ambos objetos y, posteriormente, se modifica el manifiesto escribiendo los cambios realizados mediante una variable de tipo XMLOutputter.

El objeto compuesto es validado de la misma forma que se validan los objetos que se muestran en el árbol de composición. Si el objeto es validado satisfactoriamente, se continúa con la composición. En caso contrario, se muestra un mensaje de error y no se termina la composición de ambos objetos.

En caso de continuar la composición, se copian los recursos del objeto contenido (*objetoHijo*) en la carpeta del objeto contenedor (*objetoPadre*), mediante objetos de las clases FileInputStream y FileOutputStream.

Por último se actualizará el árbol compuesto eliminando el objeto hijo (puesto que ya ha sido compuesto) y actualizando el objeto padre con los cambios realizados en el mismo.

Todos los cambios realizados hasta ahora en la composición han sido guardados únicamente en carpetas temporales. Si los cambios no son guardados posteriormente por el usuario, las composiciones realizadas no se verán reflejadas.

A.2.2.3 Secuencia "Guardar como..."

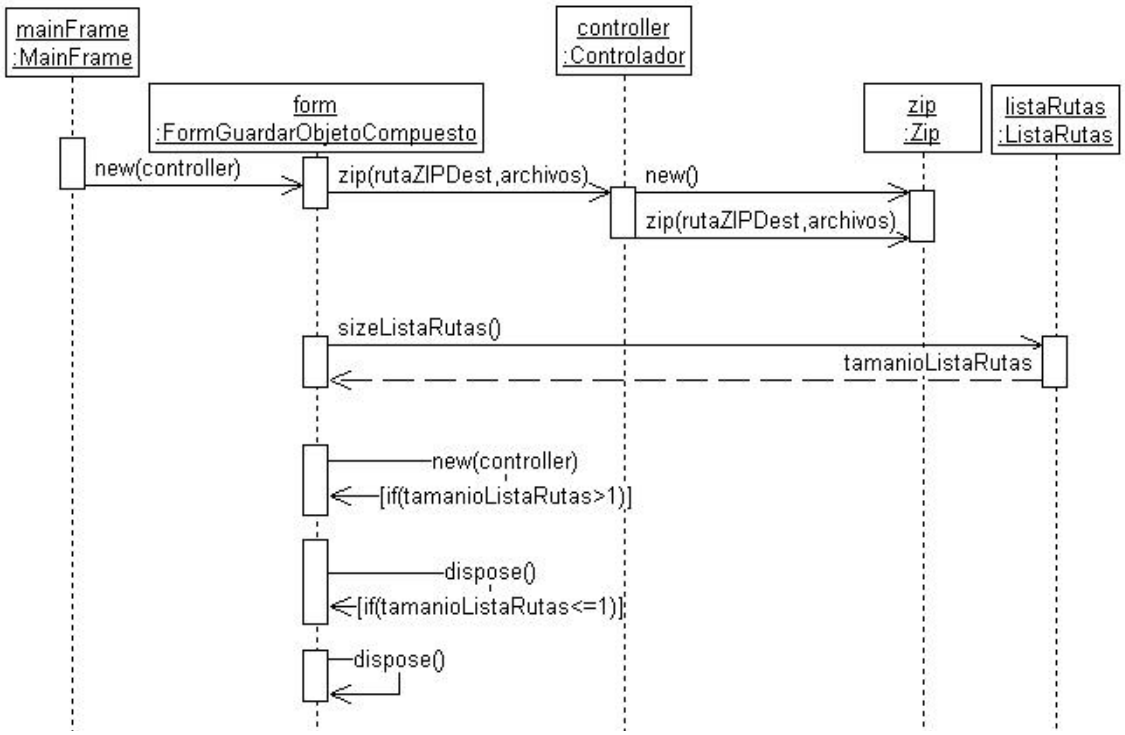


Diagrama A.2.2.3 Secuencia "Guardar como..."

En este diagrama se muestra la secuencia de pasos que sigue la herramienta LOMEditor para guardar objetos ya compuestos con el nombre dado por el usuario.

Al pulsar esta opción se comprueba que previamente se haya realizado la composición de uno ó más objetos de aprendizaje. Si esto no es así, se muestra un mensaje al usuario informándole de que no puede realizar esta acción.

En caso contrario, se muestra una ventana en la que el usuario elegirá el objeto compuesto que desea guardar.

Se almacenan en una lista (*archivos* en el diagrama de secuencia) todos los archivos que componen el objeto seleccionado por el usuario. Y se le pide la ruta destino en la que desea guardar el objeto compuesto comprimido. Con estos datos, se comprimirá el objeto.

Una vez comprimido, se muestra un mensaje en el que se informa al usuario que el objeto ha sido guardado de forma satisfactoria y se recuerda la ruta en la que se ha compuesto dicho objeto.

Si se ha compuesto más de un objeto (*if(tamanioListaRutas>1)*) se preguntará al usuario si desea guardar algún objeto más. Si la respuesta es afirmativa se mostrará de nuevo la ventana con los objetos compuestos por el usuario para que guarde de nuevo el objeto que desee.

Sin embargo, si el usuario no quiere guardar más objetos o no puede por no haber compuesto más, se cerrará la ventana.

A.2.3 Módulo 3: Módulo de evaluación de calidad

A.2.3.1 Secuencia "Mostrar Ventana evaluación manual"

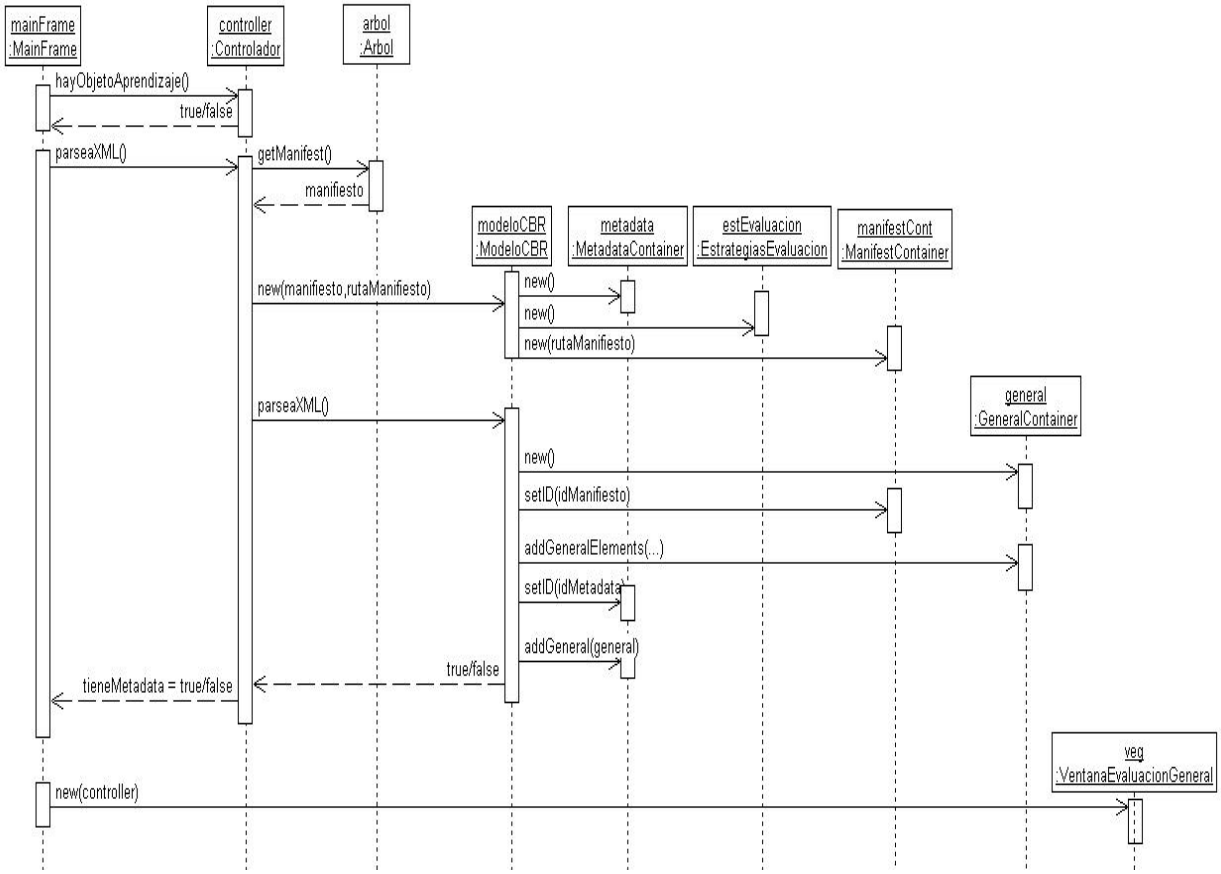


Diagrama A.2.3.1 Secuencia "Mostrar ventana evaluación manual"

Diagrama que muestra la secuencia de pasos que sigue la herramienta LOMEditor para mostrar la ventana de evaluación manual.

Como puede observarse, ante la petición del usuario, tras comprobarse que hay un objeto de aprendizaje abierto, la interfaz traspasa esta petición al controlador, el cual realiza las gestiones necesarias para llevar a cabo la operación, apoyándose en el paquete cbr.

En este caso accede al modelo cbr, desde el que generará las instancias necesarias para poder determinar los contenedores y las estrategias de evaluación adecuadas.

Finalmente podremos ver la ventana de evaluación general.

A.2.3.2 Secuencia "Mostrar resultado evaluación manual"

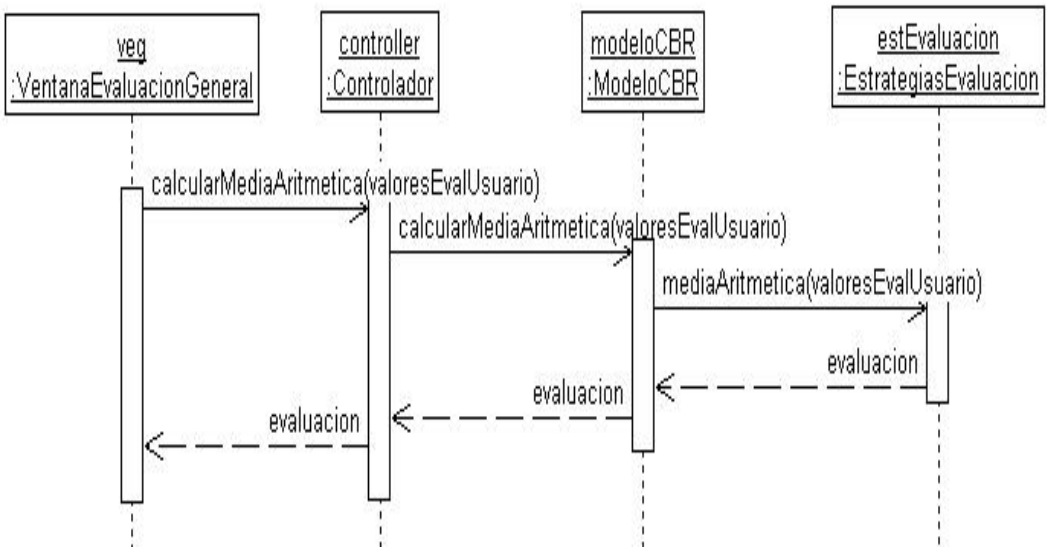


Diagrama A.2.3.2 Secuencia "Mostrar resultado evaluación manual"

Diagrama que muestra la secuencia de pasos que sigue la herramienta LOMEditor para mostrar el resultado de una evaluación manual.

Ahora vemos como se realiza la operación del cálculo de la evaluación manual, paso a paso.

En este caso será la Ventana de evaluación general la que represente a la interfaz, que transmitirá la petición al controlador. Este llamará al modelo de cbr para que complete el cálculo de la media aritmética de los valores que el usuario ha introducido como evaluación.

Finalmente se mostrará esta evaluación en la ventana.

A.2.3.3 Secuencia "Aceptar resultado evaluación manual"

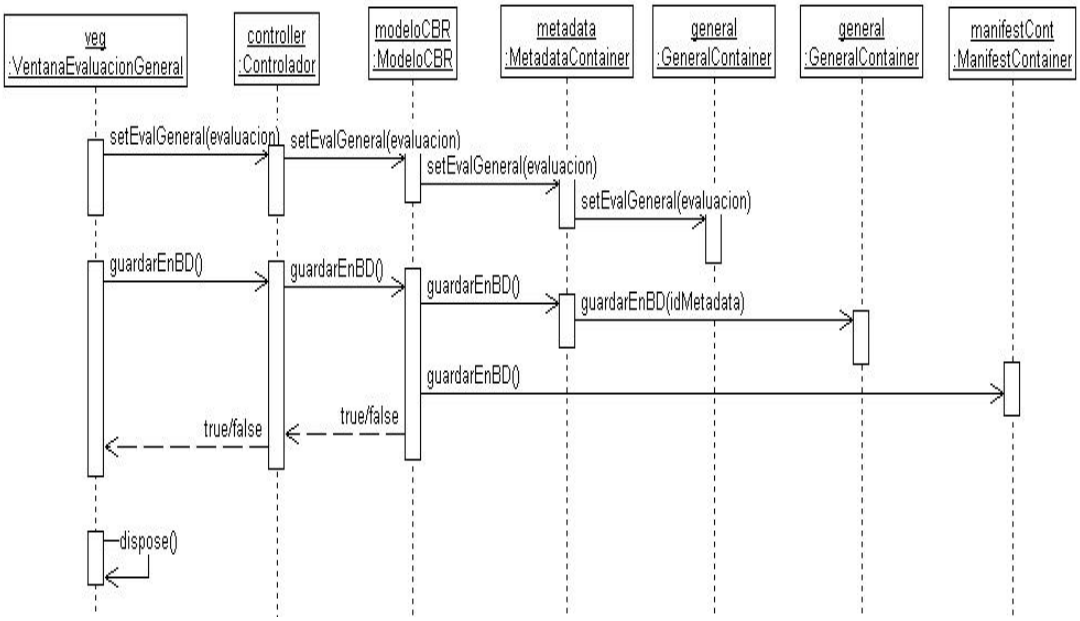


Diagrama A.2.3.3 Secuencia "Aceptar resultado evaluación manual"

Diagrama que muestra la secuencia de pasos que sigue la herramienta LOMEditor para aceptar el resultado de una evaluación manual.

Tras realizarse una evaluación manual, el usuario deberá aceptar el resultado para registrarlo en nuestro sistema.

En este caso será la Ventana de evaluación general la que represente a la interfaz, que transmitirá la petición al controlador. Este llamará al modelo de cbr para indicarle que fije la evaluación calculada para este objeto de aprendizaje.

Finalmente se guardará en la base de datos el resultado, para futuras consultas o nuevas evaluaciones.

A.2.3.4 Secuencia "Evaluación automática"

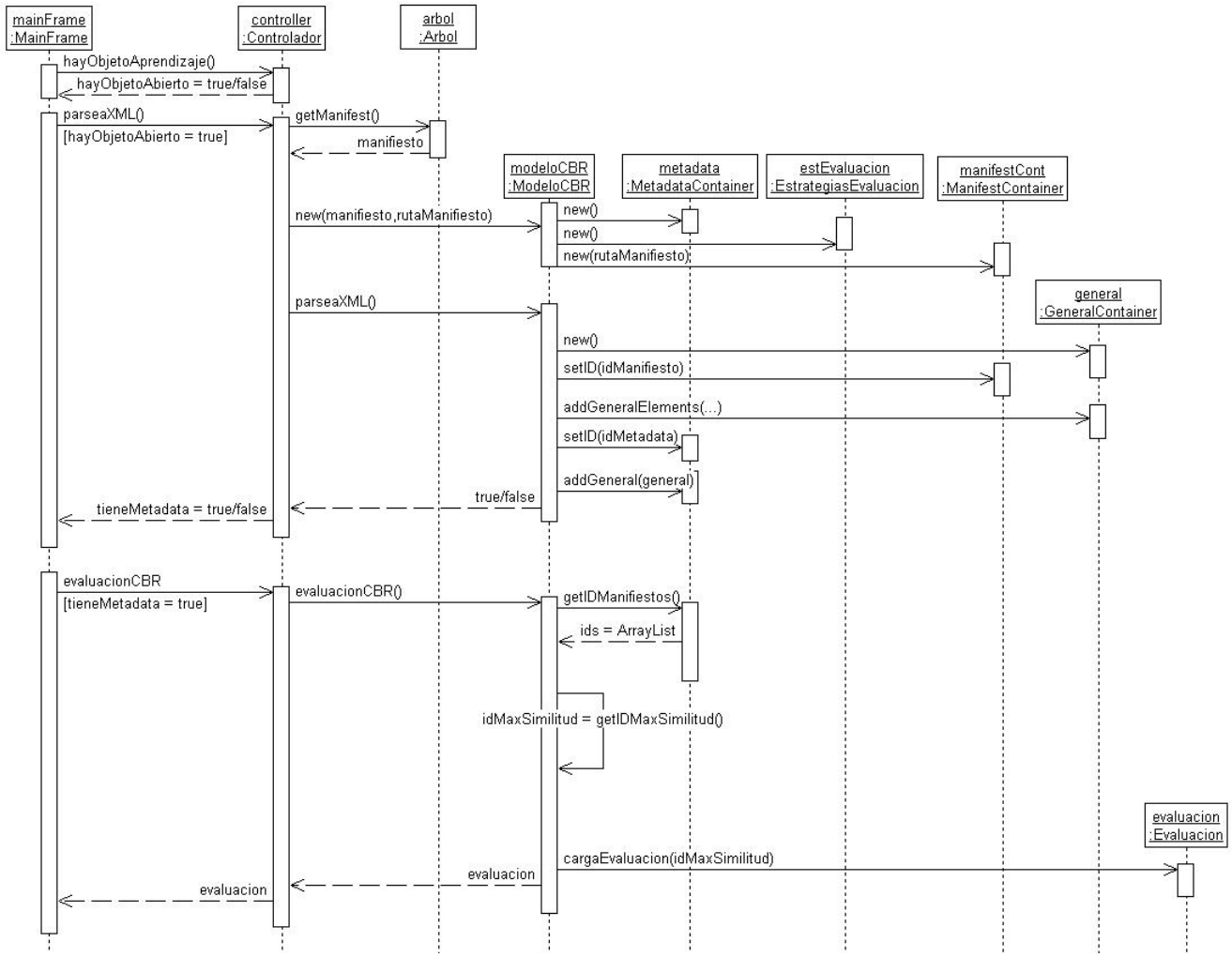


Diagrama A.2.3.4 Secuencia "Evaluación automática"

El diagrama muestra como el sistema infiere un resultado de evaluación para un objeto de aprendizaje dado, basándose en su similitud con el resto de objetos de aprendizaje que se encuentran en la base de casos del sistema.

El primer paso a realizar, como en la evaluación manual es el parsing de la estructura de los metadatos del objeto de aprendizaje y su almacén mediante clases *Container*.

Una vez almacenada la estructura de dichas clases y comprobado que el objeto contiene metadatos, se realiza la evaluación CBR propiamente dicha. Es decir, se define la similitud entre el objeto a evaluar y cada uno de los objetos que se encuentran en nuestra base de casos, y se recupera la evaluación asociada al objeto más similar al que estamos analizando.

A.3 Diagramas de bases de datos

A.3.1 Diagramas Entidad-Relación (E-R)

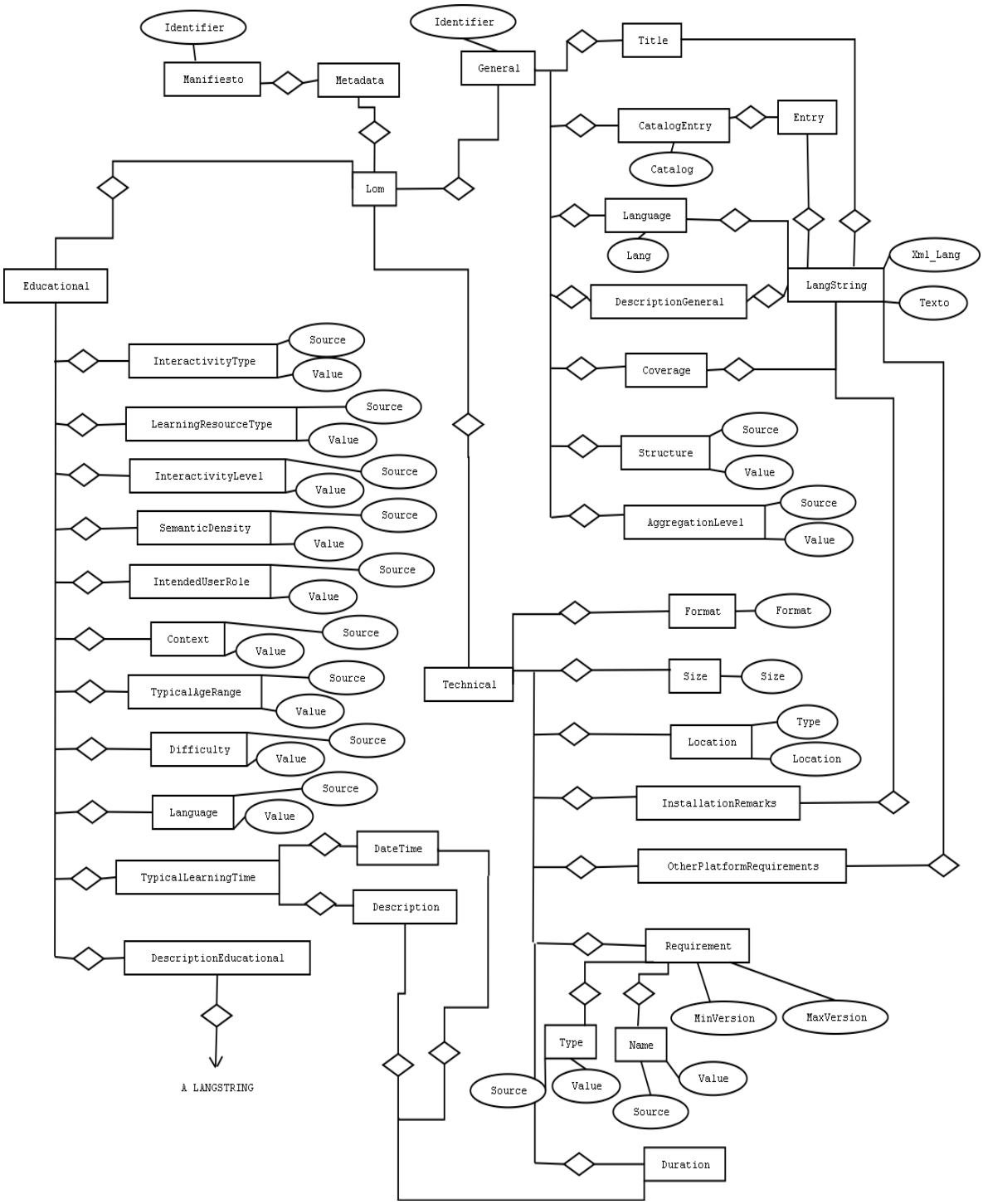


Diagrama A.3.1.- Diagrama E-R de base de datos del sistema

A.3.2 Diagramas Relacionales

DIAGRAMA RELACIONAL

GENERAL

TITLE (IDENTIFIER, XML LANG, TITLE, ELEMENTO)

KEYWORD (IDENTIFIER, XML_LANG, KEYWORD, ELEMENTO)

DESCRIPTION_GENERAL (IDENTIFIER, XML LANG, DESCRIPTION, ELEMENTO)

STRUCTURE (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

CATALOGENTRY (IDENTIFIER, CATALOG, XML LANG, ENTRY, ELEMENTO)

LANGUAGE (IDENTIFIER, LANG, ELEMENTO)

COVERAGE (IDENTIFIER, XML LANG, COVER, ELEMENTO)

AGGREGATIONLEVEL (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

EDUCATIONAL

INTERACTIVITYTYPE (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

INTERACTIVITYLEVEL (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

INTENDEDUSERROLE (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

TYPICALAGERANGE (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

TYPICALLEARNINGTIME (IDENTIFIER, DATETIME, DESCRIPTION, ELEMENTO)

LANG_USER (IDENTIFIER, LANG, ELEMENTO)

LEARNINGRESOURCETYPE (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

SEMANTICDENSITY (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

CONTEXT (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

DIFFICULTY (IDENTIFIER, SOURCE, VALUE, ELEMENTO)

DESCRIPTION_EDUCATIONAL (IDENTIFIER, XML LANG, DESCRIPTION, ELEMENTO)

EVALUACIÓN

EVALUACION (ID_MANIFIESTO, EVAL_GENERAL, EVAL_PEDAGOGICA, EVAL_TECNICA, EVAL_TOTAL)

Diagrama A.3.2.- Diagrama Relacional de base de datos del sistema

Nota: Las tablas correspondientes a EDUCATIONAL no se rellenan, han sido creadas en previsión de una futura ampliación del sistema CBR.

APÉNDICE B

MANUAL DE USUARIO

B.1 Introducción

B.1.1 Qué es IMS. Especificaciones y estándares

IMS Global Learning Consortium es una organización que se encarga de desarrollar y promover diferentes estándares y especificaciones, a nivel mundial, con el objetivo de facilitar el trabajo con e-learning.

Estos estándares ofrecen un patrón para la creación de contenidos, que soluciona el aumento descontrolado de los mismos, que termina por inutilizar muchos trabajos en el sector.

El proyecto IMS (Instructional Management System) de Global Learning Consortium, creado en 1997, es parte del W3C (World Wide Web Consortium) y está dedicado al desarrollo y promoción de arquitecturas abiertas para el aprendizaje en Internet.

Durante el auge de la industria de e-learning, la única opción disponible era trabajar con un solo proveedor de SAC. Las especificaciones inherentes del IMS rompen con este monopolio, permitiendo a las compañías hacer uso de una amplia gama de cursos sin depender en absoluto de la compañía que provea el sistema de administración del conocimiento.

LOMEditor trabaja sobre estos estándares, adaptándose a los mismos para ofrecernos la posibilidad de crear nuestros propios contenidos de enseñanza.

Si deseamos una mayor información es posible acudir al sitio Web de la organización que encontraremos en **http://www.imsglobal.org**, donde encontraremos todo lo relacionado con el tema.

B.1.2 E-Learning: Content Packaging y Metadata

Hemos mencionado anteriormente este término “e-learning”, pero ¿de que se trata? En realidad utilizamos esta expresión para referirnos al proceso de adaptación de la enseñanza a las nuevas tecnologías, especialmente Internet.

La aparición de Internet ha revolucionado la educación a distancia en todos los niveles. Por ello parece el e-learning o educación virtual, como un nuevo modo de aprendizaje, complementario al aula y, en muchas ocasiones, sustituto de la educación presencial.

Dentro de esta nueva corriente educacional encontramos diversas posibilidades, entre las que destacan los Campus Virtuales y los paquetes de contenidos u objetos de aprendizaje.

Estos últimos, los objetos de aprendizaje, son el núcleo principal de esta herramienta de autoría. Para el desarrollo de estos paquetes utilizaremos las dos principales especificaciones que IMS promueve, como son IMS Content Packaging y Metadata Specifications.

La primera comprende el conjunto de contenidos que deseamos agrupar, así como la organización que deben llevar dichos recursos. Para ello existe un manifiesto en formato "xml" que describe todo lo anterior.

La segunda, complementaria de la primera, constituye una detallada descripción del paquete de contenidos al que se refiere. Puede ocurrir también que los metadatos se refieran única y exclusivamente a una determinada parte del paquete.

Encontrarán más información al respecto en los lugares que IMS Global Consortium dedica a los mismos en su sitio Web:

http://www.imsglobal.org/content/packaging/index.cfm
y
http://www.imsglobal.org/metadata/index.cfm

B.1.3 Sobre este manual

Este documento pretende ser una guía complementaria de la herramienta LOMEditor, a través de la cual el usuario pueda recibir el soporte necesario para manejar el editor y extraer del mismo todas sus posibilidades.

Buscando introducir y familiarizar al usuario en el trabajo con objetos de aprendizaje, el manual ofrece una visión de dicho campo, a través de sus diferentes secciones, en cada una de las cuales encontraremos una detallada descripción del funcionamiento de la funcionalidad del software.

Con el fin de alcanzar el objetivo señalado, este manual se ha compuesto por las siguientes secciones:

- *Introducción a la herramienta LOMEditor*

- *Requisitos del Sistema*

- *Guía del proceso de instalación de la aplicación*

- Descripción del entorno de desarrollo

- Tutorial de manejo de la herramienta

Las imágenes de capturas de pantalla han sido tomadas a partir de LOMEditor 2005, en su versión 1.0. La estética del producto puede variar de acuerdo al entorno del sistema operativo donde se realice el trabajo.

Para mayor información sobre este producto puede dirigirse al sitio Web del proyecto, localizado en **http://usuarios.lycos.es/lomeditor/**.

B.2 LOMEditor 2005

B.2.1 Descripción general

LOMEditor 2005 es una herramienta de edición de objetos de aprendizaje, que permite por tanto la creación y modificación de Content Package y Metadata.

Se trata de una aplicación gráfica compuesta por un conjunto de servicios accesibles mediante menús y barras de herramientas, los cuales permiten realizar un completo trabajo sobre el campo anteriormente expuesto. Al tratarse de una herramienta local, no requiere de más elementos o sistemas que los incluidos en el instalador para desarrollar completamente estos paquetes de datos.

Recursos audiovisuales, textos y todo tipo de ficheros pueden ser gestionados con LOMEditor para conformar un paquete de contenidos completo y totalmente configurable. La posibilidad de edición de metadatos permite realizar una exhaustiva descripción de dicho contenido, dando así un acabado satisfactorio al objeto de aprendizaje.

LOMEditor incluye un conjunto de servicios y facilidades que hacen de la edición de objetos de aprendizaje un trabajo más sencillo. En primer lugar existe la posibilidad de evaluar la calidad del trabajo realizado, comparando de esta forma el paquete con otros similares que hayan sido tomados como referencia, o simplemente ofrece un medio para fijar objetivos a alcanzar. Por otro lado, gracias al módulo de composición de objetos de aprendizaje, el cliente puede generar objetos más completos y complejos a partir de un conjunto determinado de paquetes ya desarrollados.

Esta amplia gama de funcionalidades se unen a una ventaja y una innovación en el sector, que supone la posibilidad de trabajar con objetos basados en diferentes gramáticas, ya que LOMEditor, a diferencia de otras herramientas de edición, no obliga a la utilización de una determinada implementación de los estándares, sino que asimila la de cada objeto y adapta sus servicios a la misma.

La funcionalidad y servicios ofrecidos por el editor LOMEditor en esta versión es la siguiente:

10. Creación de objetos de aprendizaje

11. Apertura de objetos de aprendizaje ya creados

12. Edición de objetos de aprendizaje

13. Creación y modificación de metadatos

14. Visualización del contenido del objeto de aprendizaje

15. Composición de objetos de aprendizaje

16. Evaluación de la calidad del objeto de aprendizaje

Por todo ello, LOMEditor 2005 se presenta como una excelente opción a la hora de trabajar en el desarrollo de Content Package y Metadata.

B.2.2 Requisitos del Sistema

A continuación se listan los requisitos que el entorno operativo debe poseer para un correcto funcionamiento de la herramienta. Recomendamos al usuario trabajar sobre sistemas que cumplan estos mínimos, ya que en otro caso no se garantiza un comportamiento adecuado.

Requisitos Hardware mínimos

- Portátil con pantalla TFT 14.1" o mayor con área de pantalla >= 1024 x 768
ó Monitor 15" con área de pantalla >= 1024 x 768
- Espacio libre en disco: 90 MB
- Pentium® MMX® 233 o AMD® 500
- 128 MB de RAM
- Lector de CD-ROM 16x, ratón y teclado

Requisitos Hardware óptimos

- Portátil con pantalla TFT 14.1" con área de pantalla = 1024 x 768
ó Monitor 17" con área de pantalla = 1024 x 768
- Espacio libre en disco: 100 MB
- Pentium® III 700 o AMD® 1250
- 512 MB de RAM
- Lector de CD-ROM 32x, ratón y teclado

Requisitos Software mínimos

- Microsoft® Windows® 98 / 2000 / XP
- Máquina virtual de Java con JDK 1.4.2 o posterior
- Navegador Microsoft® Internet Explorer 5 o posterior

Requisitos Software óptimos

- Microsoft® Windows® XP
- Máquina virtual de Java con JDK 1.5
- Navegador Microsoft® Internet Explorer 6

Se recomienda al usuario, de cara a un correcto comportamiento del visionado de los recursos, contar con las aplicaciones necesarias para dicha apertura en caso de desear trabajar con ficheros de tipo no estandarizado.

B.2.3 El entorno LOMEditor

Para familiarizar al usuario con la utilización de la herramienta, describiremos a continuación el entorno de trabajo y desarrollo de objetos de aprendizaje LOMEditor.

En primer lugar echaremos un vistazo general al aspecto que ofrece el editor, donde podemos observar los diferentes elementos que entrarán en juego a la hora de trabajar sobre él.

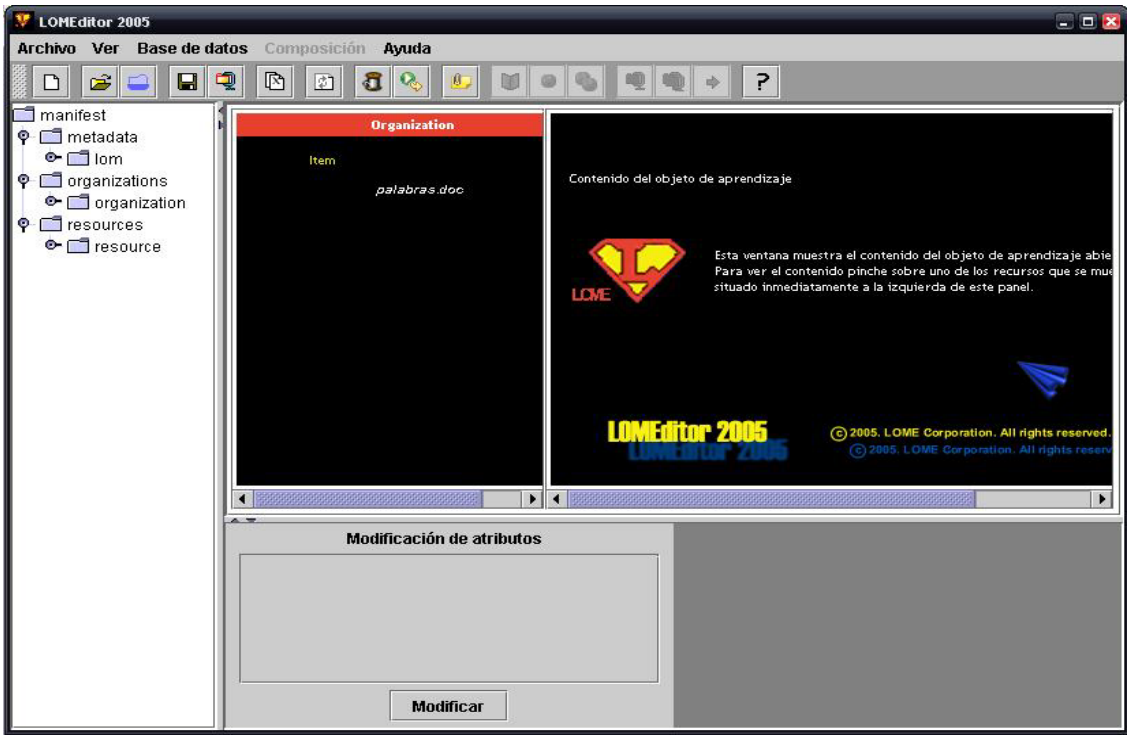


Imagen B 2.1 – Vista general aplicación

Como puede observarse, se trata de una sencilla interfaz, similar a las habituales en este tipo de herramientas, donde encontramos un menú organizado por servicios, una barra de herramientas que contiene accesos directos a estos servicios, y un conjunto de paneles de edición, cada uno de los cuales encargados de contener unos determinados elementos.

El núcleo que contiene la funcionalidad de la herramienta se encuentra situado en la parte superior de la herramienta, como se puede observar en la imagen siguiente.

Imagen B 2.2 – Menú principal y barra de herramientas

Ahora vamos a ver más detalladamente todos estos componentes, que nos servirán para acceder a la diversa funcionalidad que ofrece LOMEditor.

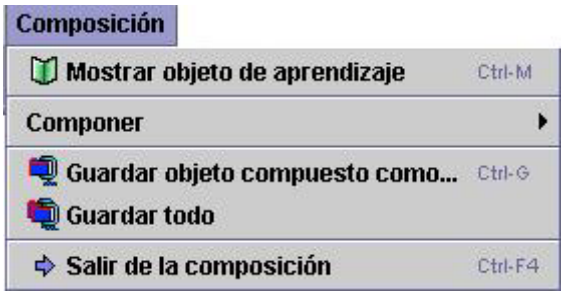
1. El menú principal

En la parte superior de la ventana encontramos este menú, donde se encuentran organizadas todas las capacidades del editor. Vamos a ver detalladamente cada una de ellas:

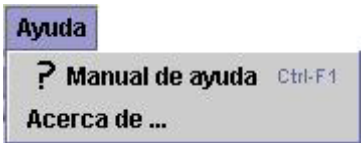
Este menú contiene las habituales operaciones sobre ficheros o archivos. Encontramos entre ellas “Nuevo”, “Abrir”, “Guardar”, “Cerrar objeto”, y “Salir”.

En este menú disponemos de la opción de acceso a la visualización del contenido del objeto de aprendizaje. Permite refrescar el mostrado automático que ofrece la herramienta.

Este otro menú lista las opciones de trabajo sobre la Base de datos. Por un lado la parte de CBR, donde realizaremos las evaluaciones de calidad, ya sean automáticas o manuales. Por otro lado nos encontramos con las consultas directas a la base de datos, donde recuperar los objetos de aprendizaje almacenados.



Para el módulo de composición existe este menú, donde localizar las diversas opciones que dicho servicio presenta. Los botones que permiten ir abriendo y mostrando objetos de aprendizaje para la composición, así como el guardado de los mismos, junto con la opción de componer, donde encontraremos la posibilidad de composición en paralelo o profundidad. Finalmente disponemos del botón de salida del modo composición.



Menú donde encontrar el manual de ayuda y la información corporativa referente a LOMEditor 2005.

2. La barra de herramientas

Dentro de la aplicación LOMEditor existe, bajo el menú principal, una barra de herramientas que contiene botones de acceso directo a la principal funcionalidad de la herramienta. El contenido de la misma es el siguiente:

 Nuevo: Crea un nuevo objeto de aprendizaje

 Abrir: Abre un objeto de aprendizaje almacenado en disco

 Guardar: Guarda los cambios realizados en el objeto de aprendizaje

 Comprimir: Guarda los cambios y comprime el objeto de aprendizaje en un paquete con extensión zip.

 Cerrar: Cierra el objeto de aprendizaje que se encuentre actualmente abierto en el editor.

 Composición: Habilita el modo composición, abriendo un objeto para dicho fin.

 Composición en paralelo: Compone en paralelo dos objetos de aprendizaje

 Composición en profundidad: Compone dos objetos en paralelo.

 Guardar objeto en composición: Guarda el objeto que se encuentre actualmente en composición.

 Guardar todos los objetos en composición: Guarda todo objeto abierto en el modo composición.

 Abandonar composición: Permite salir de la composición de objetos.

 Evaluación manual: Acceso a la evaluación manual de la calidad del objeto de aprendizaje abierto.

 Evaluación automática: Acceso a la evaluación automática de la calidad.

 Consulta a la base de datos: Consulta a la base de datos de objetos de LOMEditor.

 Ayuda: Acceso a la ayuda que proporciona el editor.

3. *El panel de modificación*

Este panel es utilizado durante la modificación de elementos del manifiesto del objeto de aprendizaje. En él se listan los atributos modificables, junto con un campo editable para este fin; contiene a su vez un botón para registrar los cambios realizados.

Imagen B 2.3 – Panel de modificación

4. *Vista en árbol*

Este panel, situado en el lado izquierdo de la herramienta, contendrá una vista en árbol del manifiesto del objeto de aprendizaje abierto en ese momento. Cualquier labor de edición debe ser realizada sobre él.

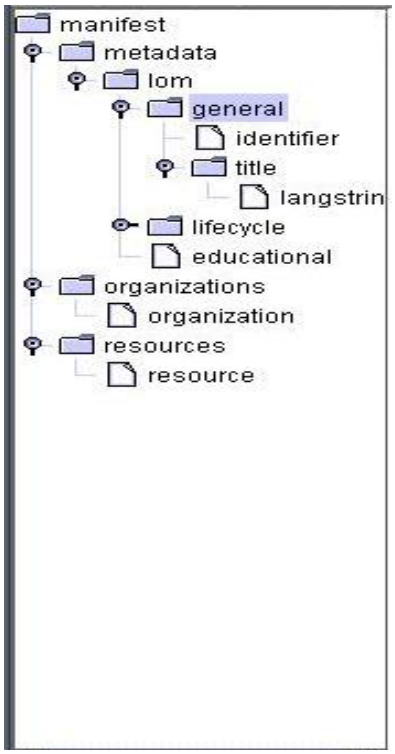


Imagen B 2.4 – Panel de visualización y edición del árbol

5. Panel de visualización

En este panel mostrará LOMEditor los recursos incluidos en el objeto de aprendizaje.

Imagen B 2.5 - Icono del instalador

6. Panel de información

Sencillo panel, que encontraremos en la esquina inferior derecha, donde se puede visualizar información acerca del nodo seleccionado.

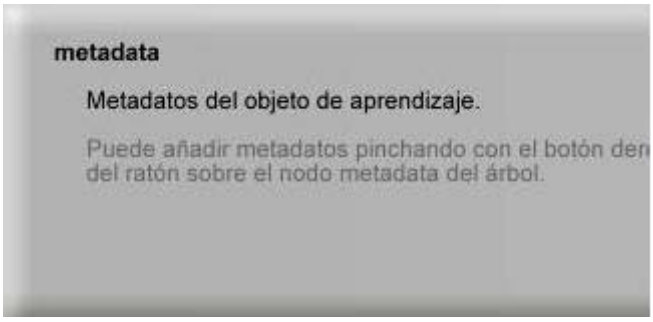


Imagen B 2.4 - Icono del instalador

B.3 Tutorial de Instalación

Para facilitar la instalación de LOMEditor 2005 se entrega un instalador automático del software necesario. Este instalador solo necesita únicamente ser ejecutado en un entorno operativo que cumpla los requisitos mínimos anteriormente expuestos.

Para instalar LOMEditor debemos ejecutar este instalador en nuestra máquina, como se describe a continuación:

En primer lugar, una vez en posesión del ejecutable Instalador de LOMEditor 2005, debemos pulsar con doble click sobre él.

Imagen B 3.1 - Icono del instalador

Al hacerlo veremos aparecer una ventana de diálogo, a través de la cual realizaremos todo el proceso de instalación. Lo primero que debemos hacer es seleccionar el idioma. En esta primera versión se dispone únicamente de español.

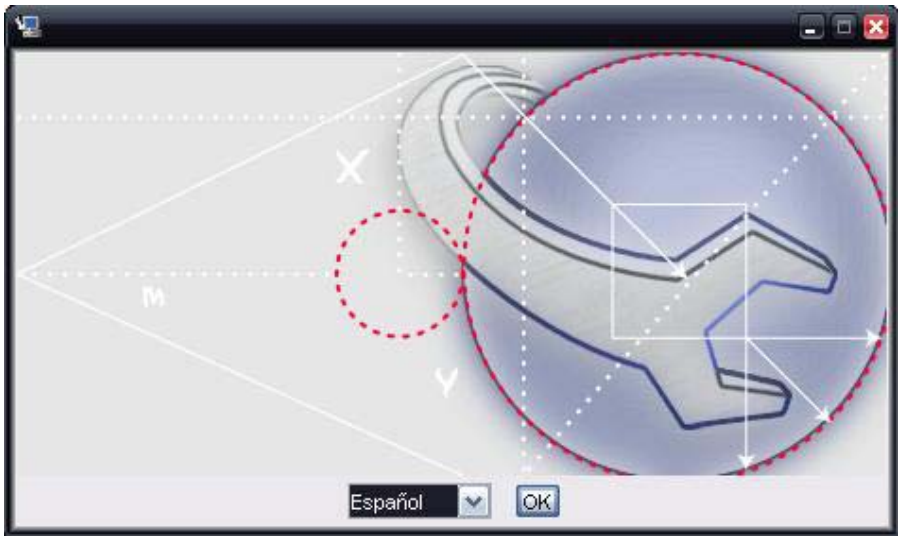


Imagen B 3.2 – Selección de idioma

Pulsando sobre el botón OK continuaremos la instalación seleccionada.

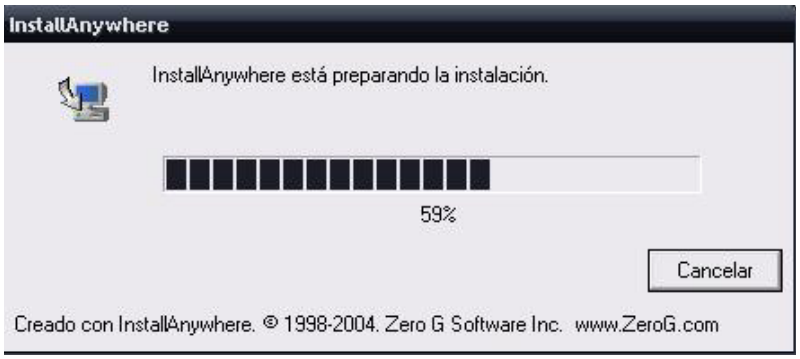


Imagen B 3.3 - Icono del instalador

Cuando en instalador termina de arrancar veremos aparecer de nuevo el cuadro de diálogo que nos va a seguir guiando en este proceso. Como puede observarse en la siguiente imagen, encontramos ya listados los diferentes pasos a seguir para completar la instalación.

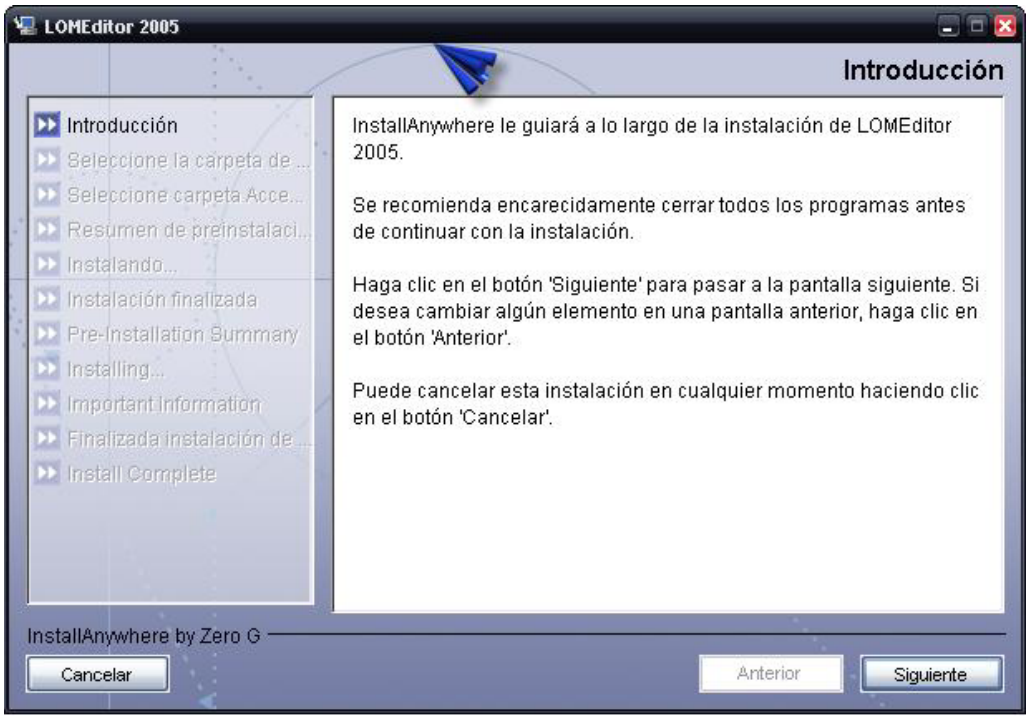


Imagen B 3.4 - Introducción de la instalación

De nuevo, como sucederá a partir de ahora, debemos pulsar el botón siguiente para proseguir la instalación. Si en algún momento deseamos abortar el proceso, el botón cancelar nos permitirá hacerlo.

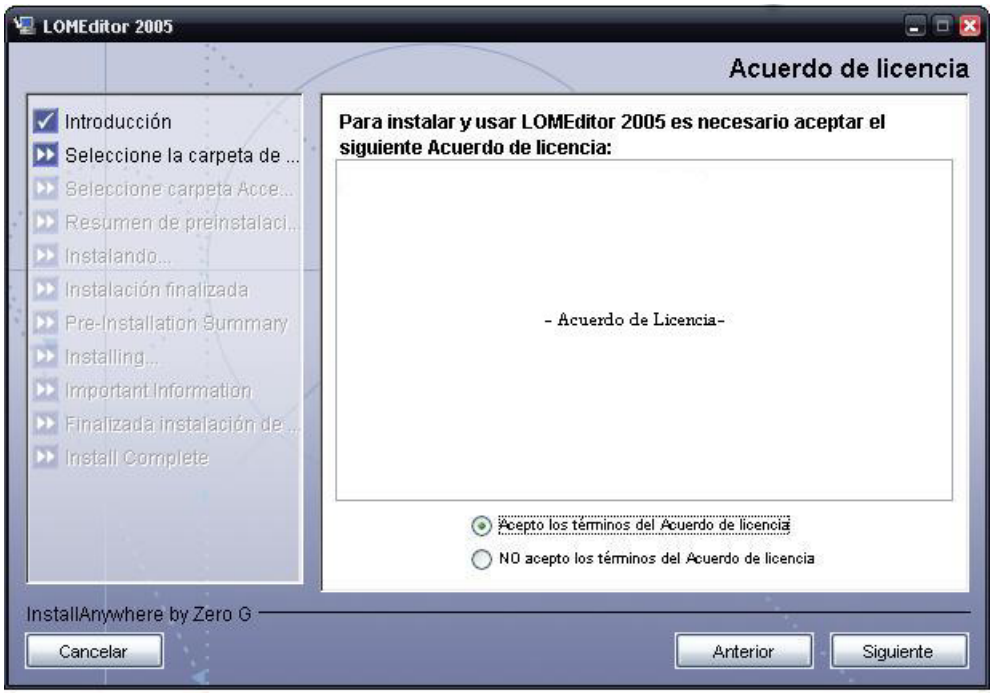


Imagen B 3.5 - Acuerdo de licencia

Pasado este punto encontraremos el acuerdo de licencia. Para que se nos permita acceder al siguiente paso debemos leerlo y aceptar sus términos.

Imagen B 3.6 - Selección de máquina virtual

Prestaremos gran atención en este punto, ya que es el momento de seleccionar la máquina virtual Java que deseamos utilizar. Windows provee de una por defecto, pero si deseamos cambiarla por una más actual, podremos hacerlo mediante el botón "Seleccionar otra…", donde exploraremos el sistema en busca de la misma. Es importante indicar una máquina virtual válida, ya que LOMEditor requiere de la misma para su ejecución.

Imagen B 3.7 - Selección de paquete de instalación

Indicamos ahora el conjunto de instalación que deseamos. Se recomienda marcar el primer conjunto, ya que resulta más completo.

Imagen B 3.8 - Selección de ubicación

Tras estos pasos llega el momento de indicar al instalador el lugar exacto donde vamos a realizar la instalación. Por defecto LOMEditor se situará junto al resto de aplicaciones del sistema.

Imagen B 3.9 – Acceso directo a la herramienta

Finalmente tenemos que seleccionar algunos parámetros más, como son la forma en la que deseamos acceder en un futuro a la herramienta. Una opción interesante es la de crear estos accesos para todos los usuarios del equipo, en caso de estar este compartido.

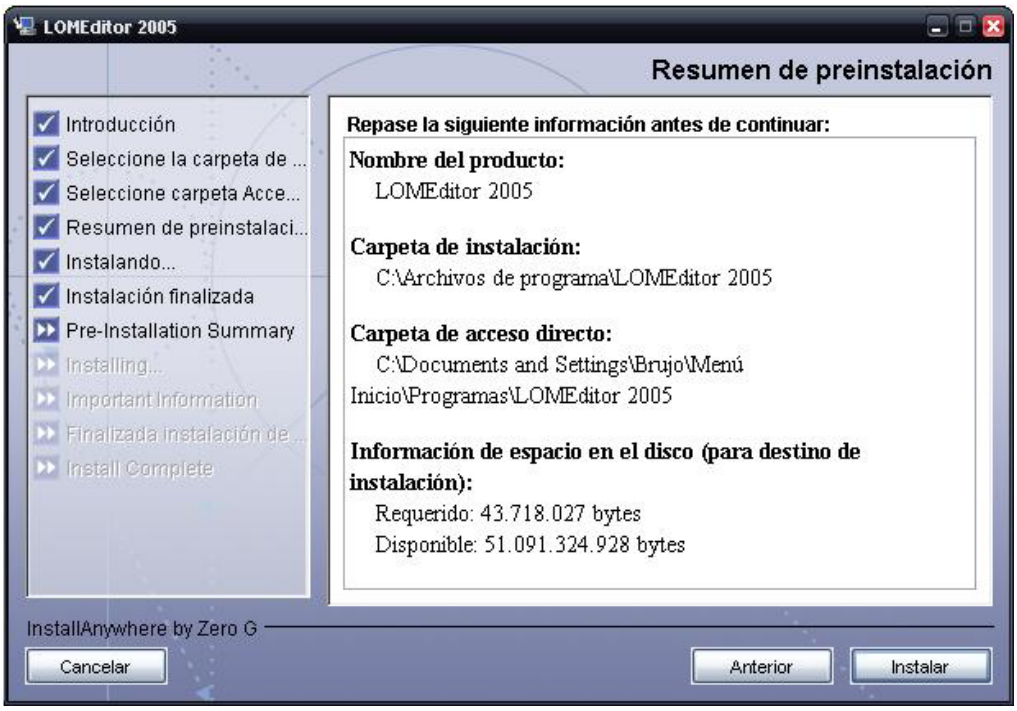


Imagen B 3.10 – Resumen de instalación

Como medida de seguridad se muestra un listado con la información resumen de la instalación. Si encontramos algún aspecto con el que no estamos de acuerdo, es el momento de volver atrás y cambiarlo.

Si decidimos seguir, será el momento en el que el instalador procederá a descomprimir los ficheros e incorporarlos a nuestro sistema, de acuerdo con los parámetros anteriormente indicados.

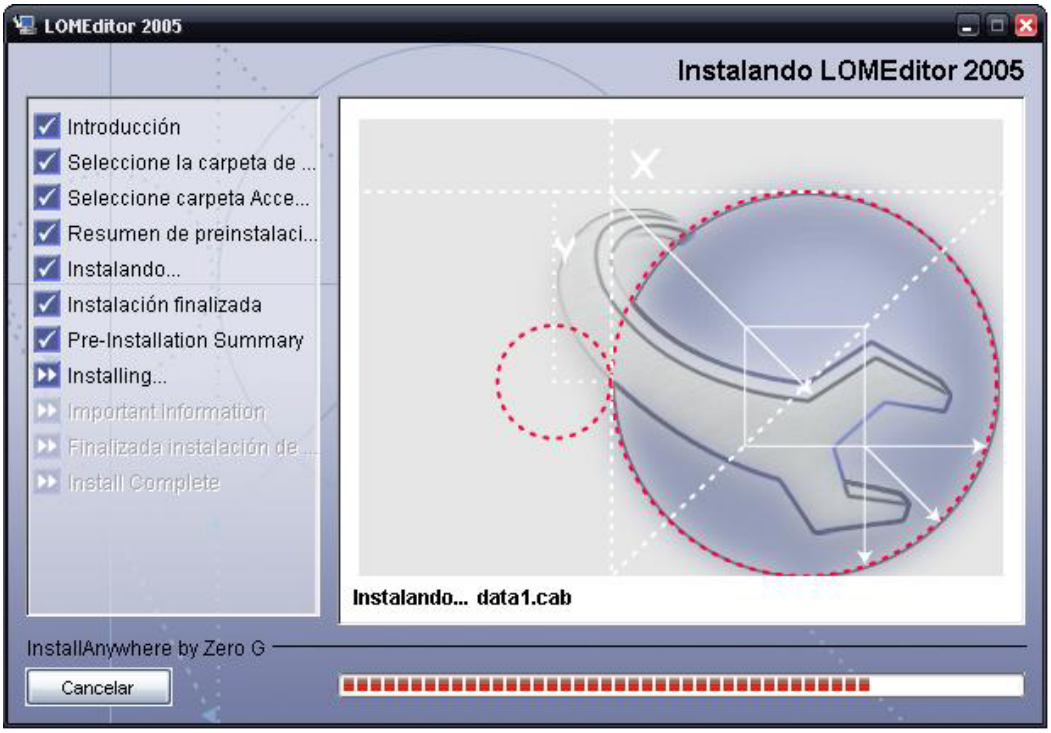


Imagen B 3.11 – Proceso de instalación de LOMEditor

Se recomienda no realizar otras operaciones en el sistema que conlleven una gran carga, pues podrían perjudicar la correcta instalación de esta herramienta.

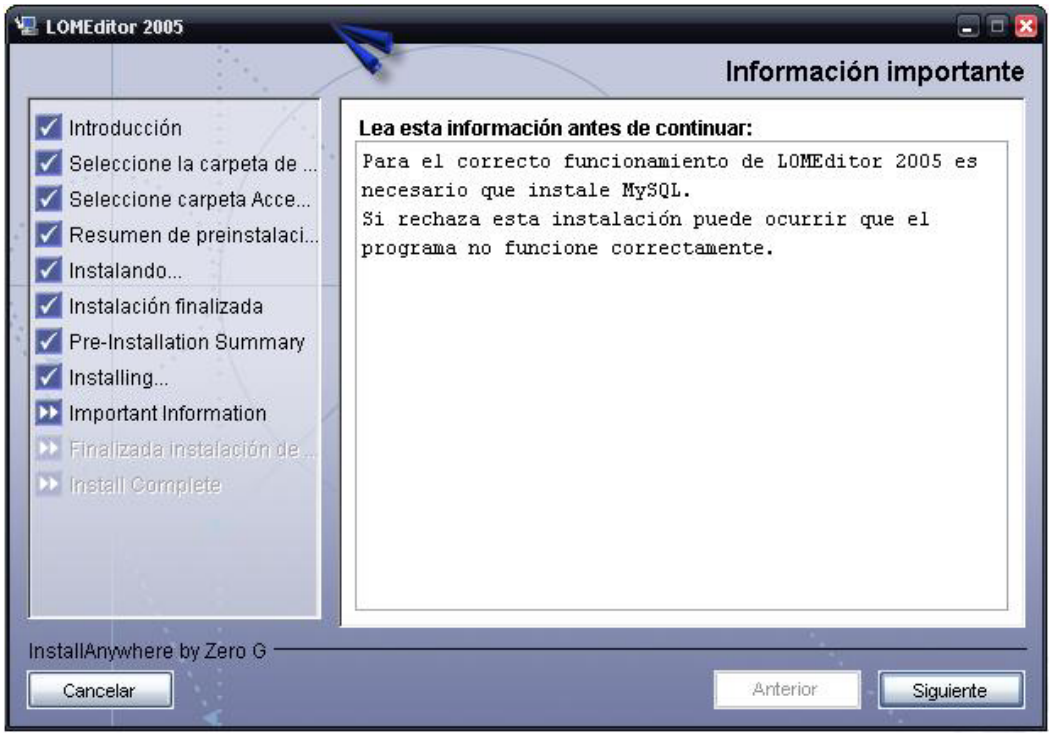


Imagen B 3.12– Información sobre MySQL

Tras instalar la aplicación seremos avisados de la necesidad de instalar MySQL, un servidor libre de bases de datos. Esta instalación es obligatoria para el correcto funcionamiento de LOMEditor 2005, ya que rechazándola no podremos acceder a ciertas funcionalidades importantes de la aplicación.

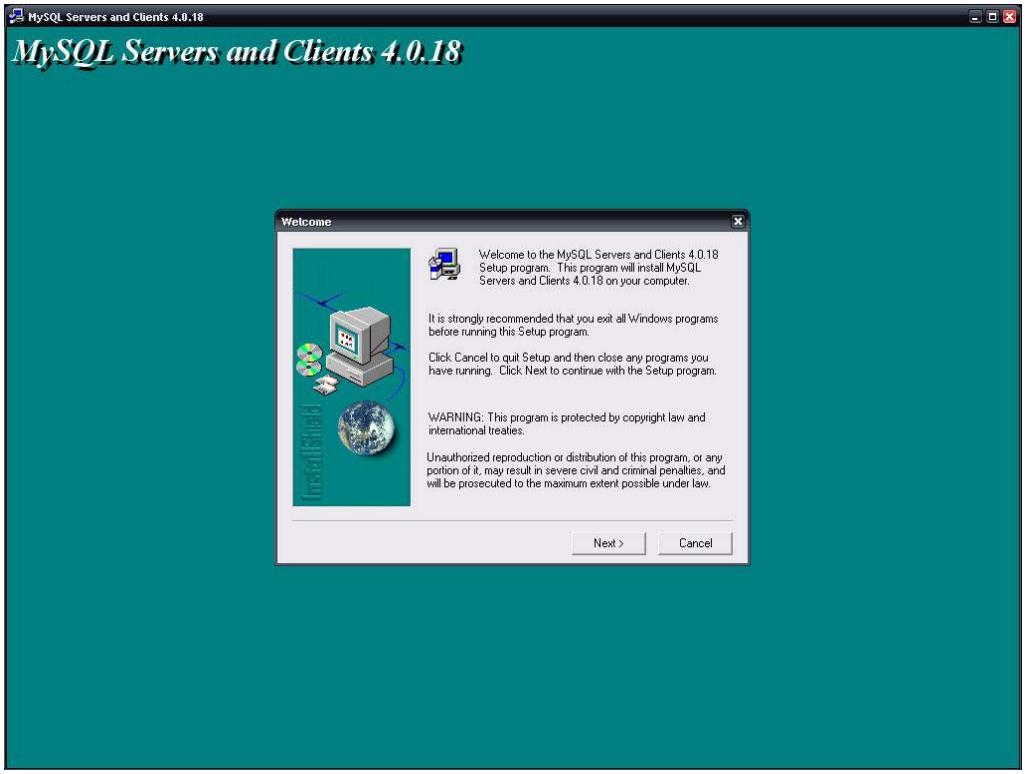


Imagen B 3.13 – Presentación del instalador de MySQL

MySQL posee su propio instalador, por el que no debemos preocuparnos, ya que se ejecuta automáticamente. Los pasos a seguir son muy sencillos en este caso; deberemos pulsar sobre el botón “Next” (siguiente) para navegar por las ventanas de diálogo que se nos van mostrando. Se recomienda mantener las opciones por defecto de la instalación de MySQL.

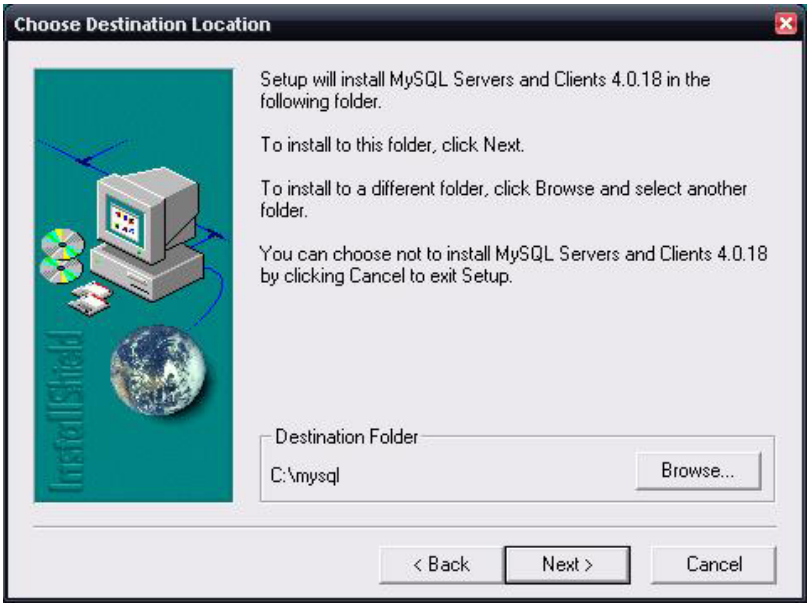


Imagen B 3.14 – Ubicación de MySQL

No obstante, si así lo deseamos, podemos configurar algunos aspectos del servidor de base de datos, entre ellos la localización de la carpeta destino.

Podremos también seleccionar el tipo de configuración a introducir.

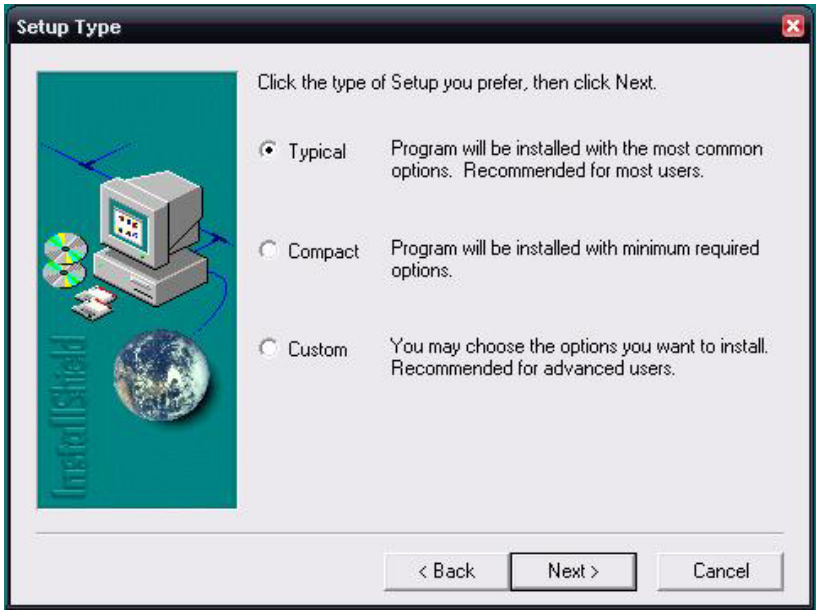


Imagen B 3.15 – Paquete de instalación de MySQL

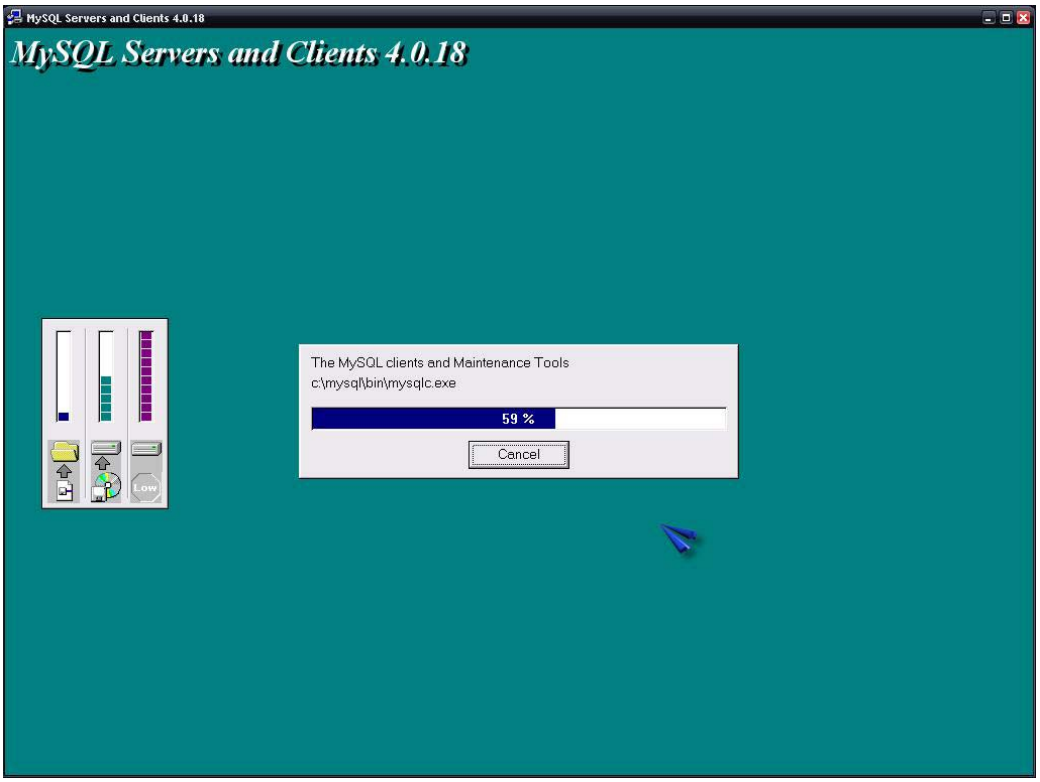


Imagen B 3.16 – Proceso de instalación de MySQL

Cuando la instalación de MySQL concluya, volveremos al cuadro de diálogo principal de la instalación de LOMEditor, donde nos informarán del estado del proceso.

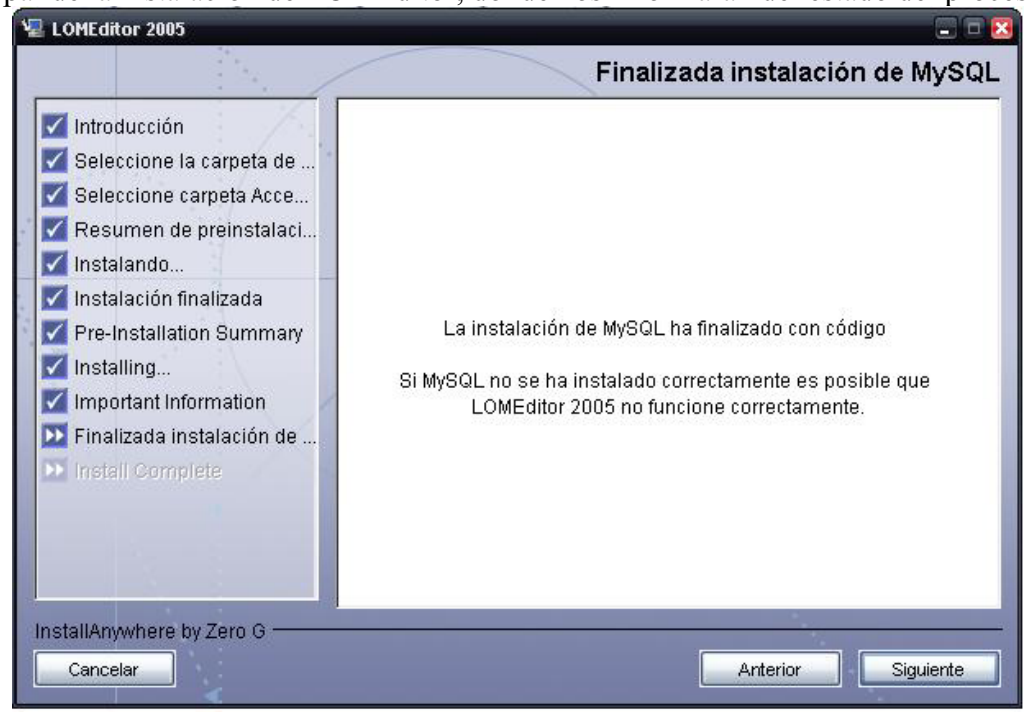


Imagen B 3.17 – Finalización de la instalación

Daremos ahora los últimos pasos para cerrar definitivamente este proceso. Para continuar, volvemos a pulsar sobre siguiente una vez más.

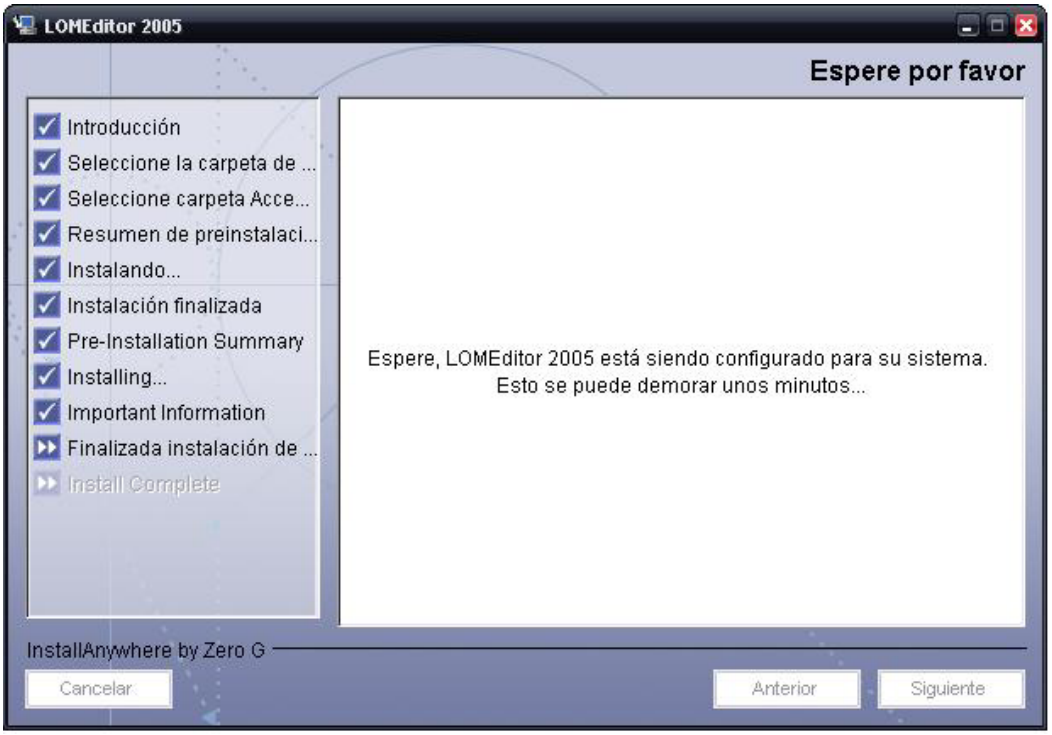


Imagen B 3.18 – Configuración de LOMEditor

Esperaremos a que LOMEditor termine de configurarse y enlazarse con la base de datos y los diferentes elementos que intervendrán en su ejecución. Se recomienda nuevamente no realizar tareas que constituyan una fuerte carga para el sistema, pues podría afectar a la configuración de la herramienta.

Imagen B 3.19 – Finalización completa del proceso de instalación

Finalmente llegaremos a este último punto del proceso de instalación de la herramienta. El instalador nos mostrará el resultado de dicho proceso, y a partir de entonces podremos disfrutar de LOMEditor.

En caso de problemas recomendamos al usuario desinstalar LOMEditor y volver a reinstalarlo.

B.4 Tutorial de manejo

La siguiente sección del manual comprende los aspectos más directamente relacionados con el manejo de la herramienta.

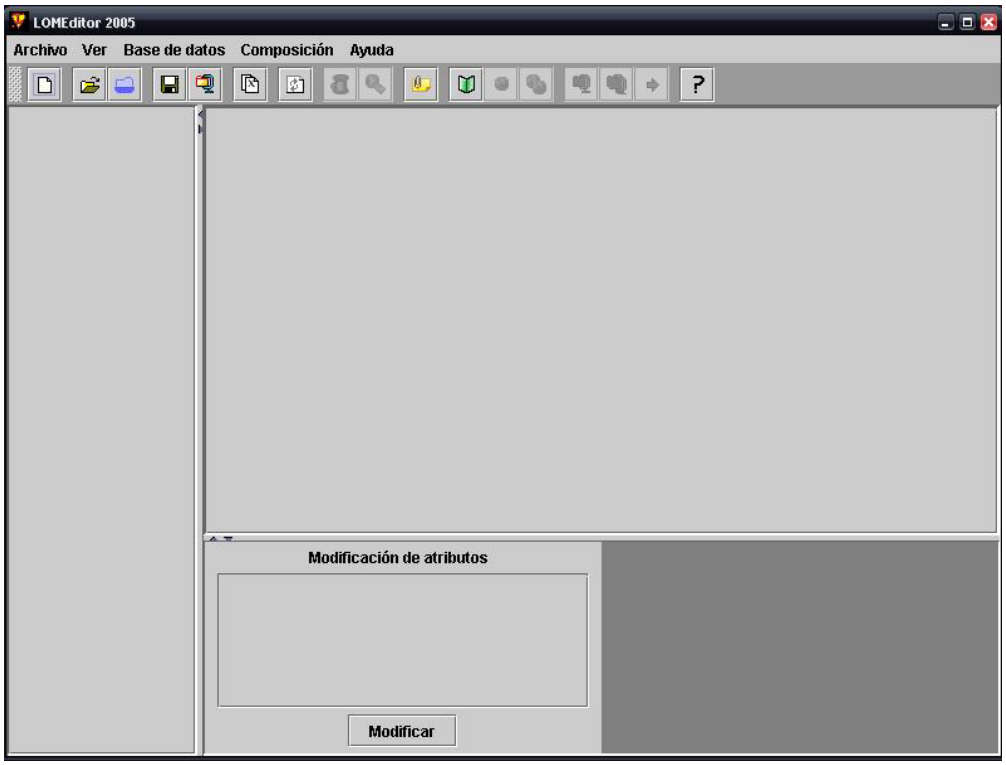


Imagen B 4.1 – Visión general de la herramienta

B.4.1 Edición de objetos de aprendizaje

Este primer apartado presenta la funcionalidad básica de LOMEditor, en lo que se refiere a sus capacidades como editor de objetos de aprendizaje.

1. Crear un nuevo objeto de aprendizaje

Para crear un nuevo objeto de aprendizaje debemos acceder a la opción "nuevo" que encontraremos en el menú, o en la barra de herramientas. Al enviar esta petición a la herramienta, esta nos requerirá, antes de generar el nuevo objeto de aprendizaje, que le indiquemos la ubicación del mismo.

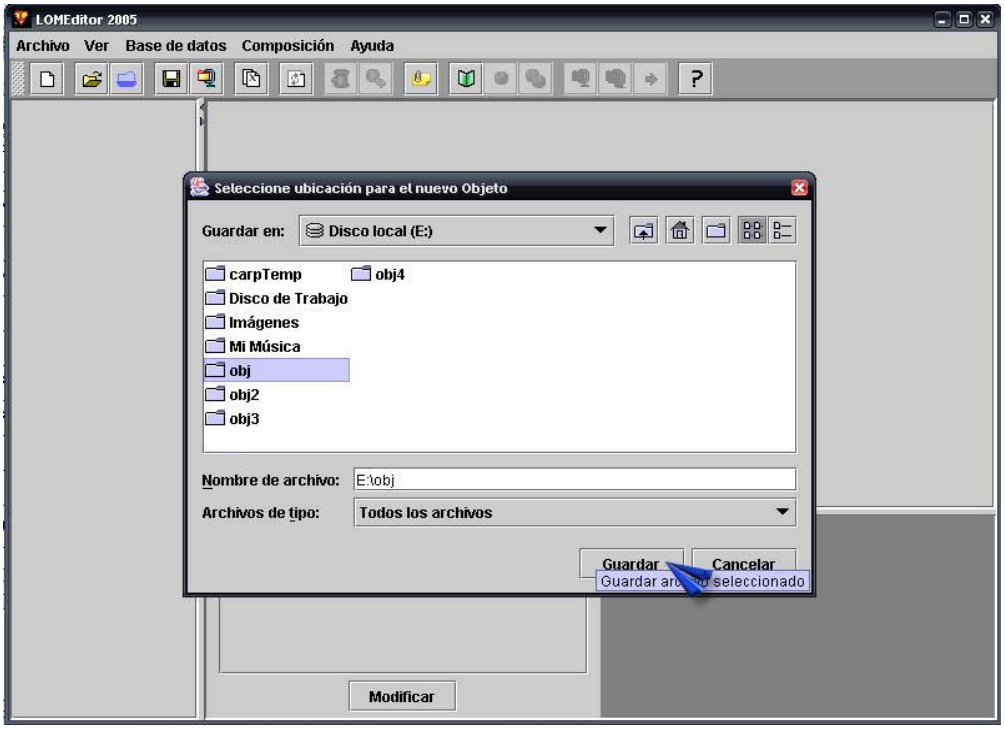


Imagen B 4.1.1 – Ubicación de nuevo objeto de aprendizaje

Mediante el cuadro de diálogo que observamos en la imagen podremos explorar el sistema y decidir el destino que más nos convenga. Una vez realizada esta operación, LOMEditor abrirá el objeto para que podamos comenzar el trabajo con el mismo.

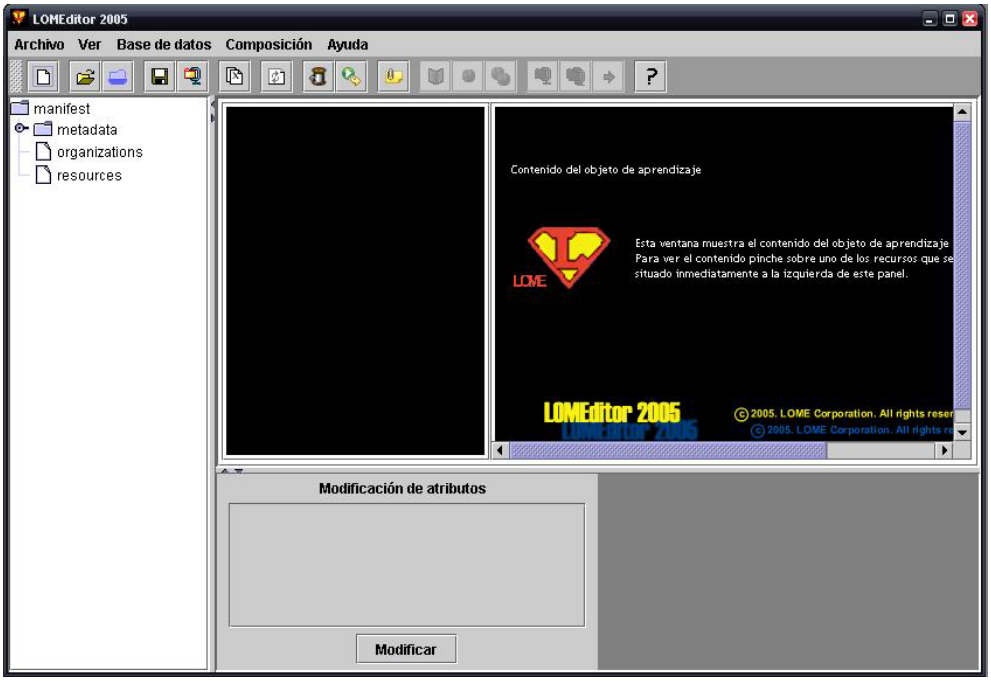


Imagen B 4.1.2 – Visión objeto abierto

En los siguientes puntos veremos como trabajar sobre los objetos abiertos en nuestro editor.

2. *Guardar un objeto de aprendizaje*

Existen dos formas de guardar un objeto de aprendizaje. Ambas son muy sencillas.

En el primer caso únicamente guardaremos las últimas modificaciones realizadas sobre el objeto de aprendizaje. Esto hará que LOMEditor guarde en el fichero físico que contiene la información del objeto de aprendizaje dichos cambios. Para esto simplemente deberemos pulsar sobre el botón del menú o de la barra de herramientas destinado a este fin.

Para guardar el objeto de aprendizaje, y comprimirlo en forma de paquete accederemos a otro servicio de la herramienta, claramente identificado por el icono de compresión habitual.

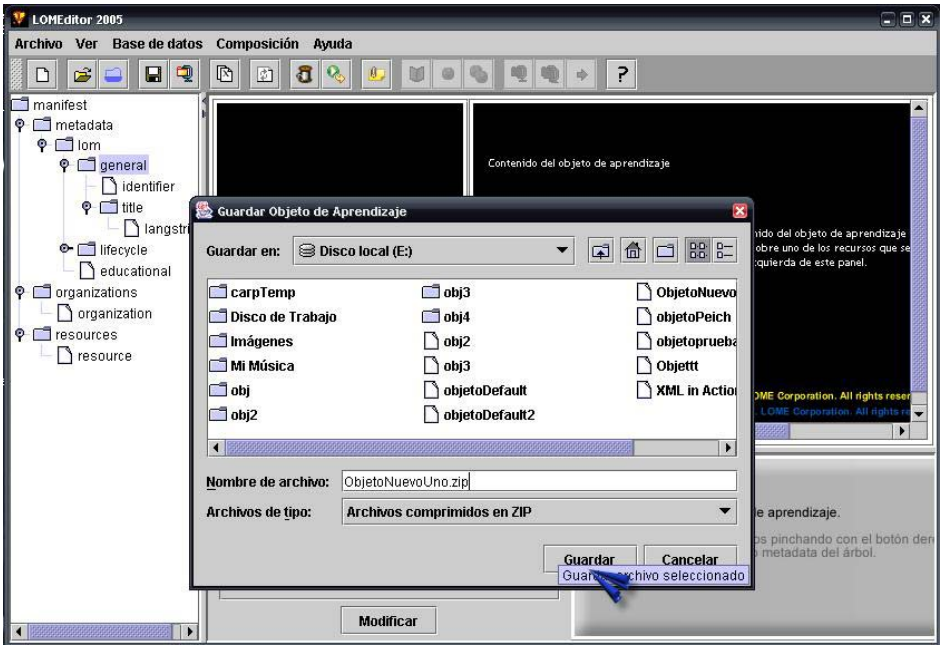


Imagen B 4.1.3 – Guardar objeto de aprendizaje

Esta vez el sistema nos pedirá el nombre y destino del fichero .zip, que contendrá el paquete del objeto de aprendizaje. Este fichero será nuevamente tratable desde la herramienta (Ver el punto “Abrir un objeto ya creado”).

3. Abrir un objeto ya creado

Esta es una funcionalidad muy importante dentro del editor, ya que nos permite recuperar objetos no solo creados anteriormente por nosotros mismos, sino muchos otros generados por otras herramientas, e incluso manualmente.

Estos objetos deben contener, como premisa obligatoria, una gramática que constituya el patrón de creación del objeto, y un manifiesto basado en la misma, y correctamente construido. Esto es así porque LOMEditor asimila dicha gramática y permite no solo abrir, sino también editar, todo tipo de objetos bien formados.

Teniendo este punto claro, para abrir un objeto de aprendizaje accederemos al servicio mediante la interfaz de usuario, pulsando sobre el botón de abrir, ya sea desde el menú o desde la barra de herramientas.

En las siguientes figuras podemos ver, una vez más, un ejemplo de acceso a los servicios de LOMEditor mediante su interfaz.

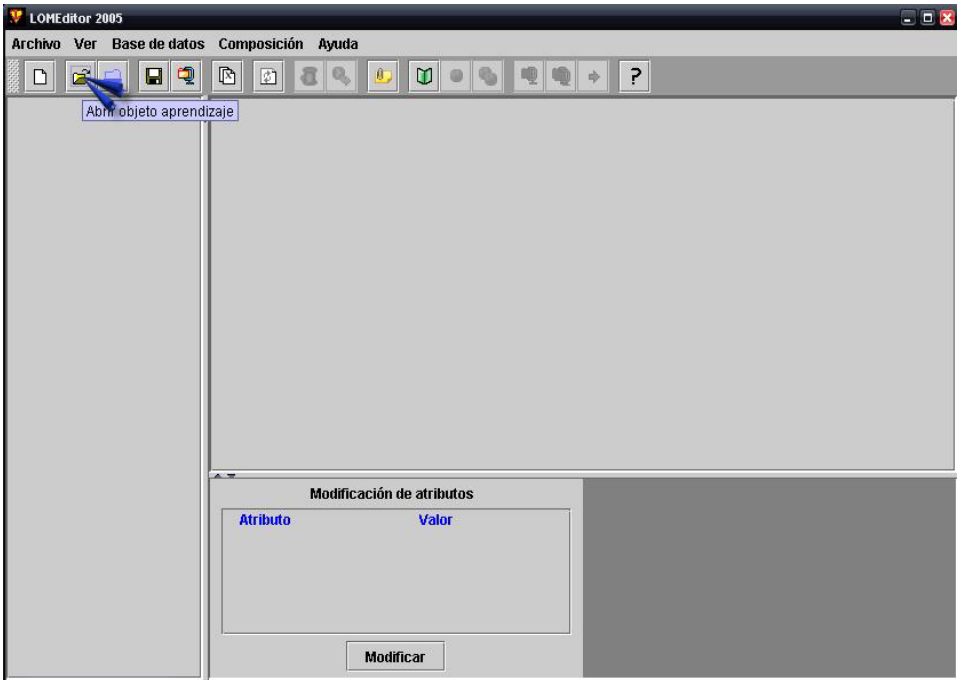


Imagen B 4.1.4 – Botón Abrir barra herramientas

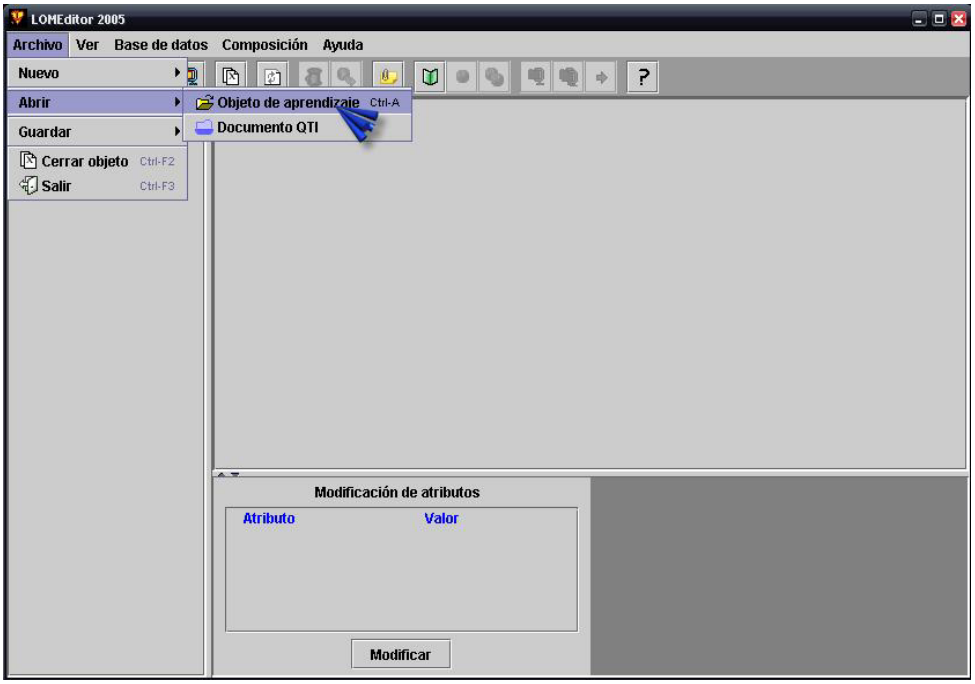


Imagen B 4.1.5 – Botón Abrir menú principal

A diferencia de la creación de nuevo objeto, esta vez, lógicamente, deberemos indicar al sistema el recurso elegido para su apertura. Para ello recurrimos de nuevo a los cuadros de diálogo.

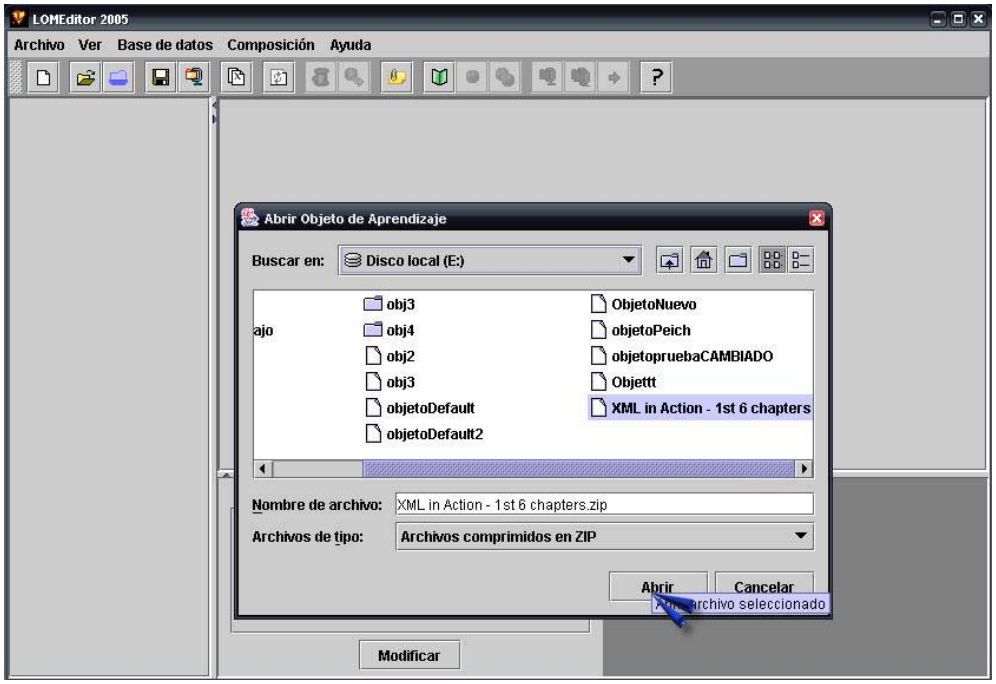


Imagen B 4.1.6 – Cuadro de diálogo apertura objeto

Una vez elegido el objeto, de nuevo buscaremos un destino para el mismo, análogamente a como lo hemos hecho para crear un objeto nuevo.

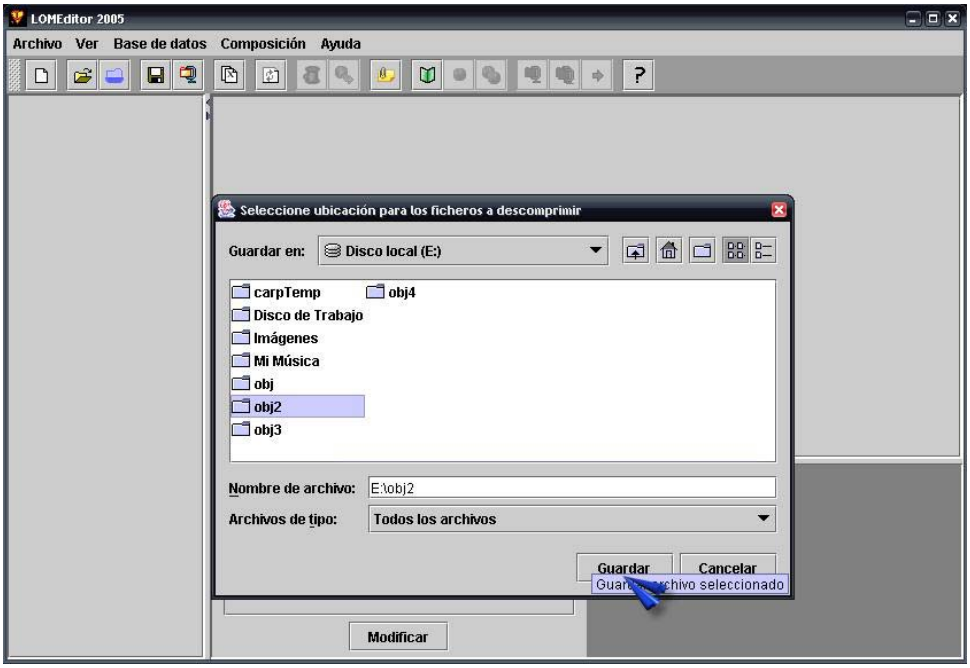


Imagen B 4.1.7 – Botón buscar destino

Finalmente logramos el objetivo que nos proponíamos, abrir un objeto anteriormente almacenado en nuestro editor, para poder trabajar con él si lo deseamos.

4. Editar un objeto abierto

Una vez que tenemos un objeto abierto en la herramienta, ya sea nuevo o recuperado de una edición anterior, estamos en disposición de trabajar sobre él.

En primer lugar vemos como podemos crear nuevos nodos en el árbol que representa el objeto de aprendizaje. Se hará de igual forma tanto para Resources, Organizations como Metadata.

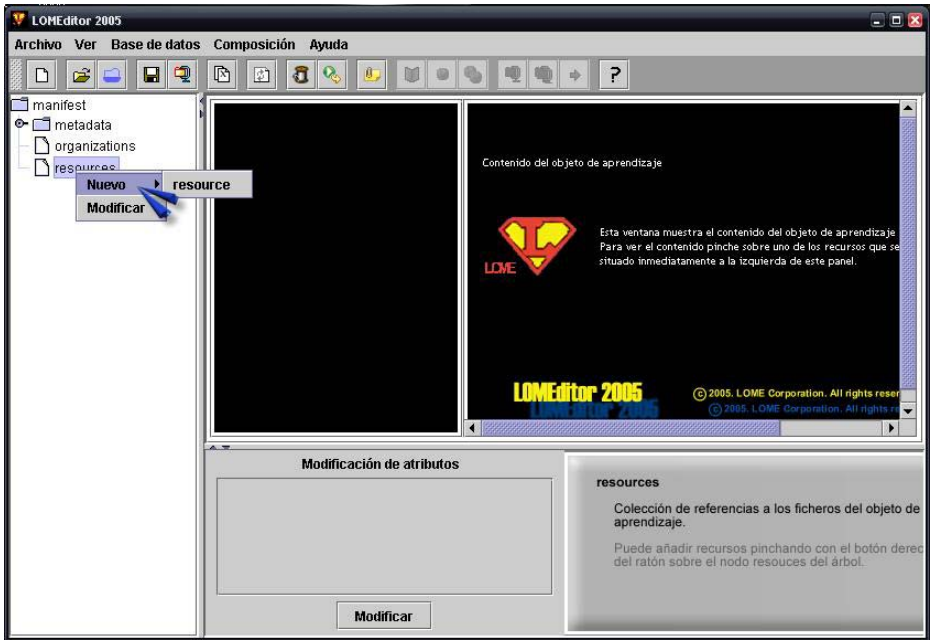


Imagen B 4.1.8 – Ejemplo edición objeto

El proceso es sencillo. Pulsamos sobre el nodo donde deseamos actuar con el botón derecho del ratón. Se desplegará un submenú donde encontramos las tres operaciones principales de edición (insertar un nuevo elemento, eliminarlo o modificarlo).

Habrá elementos sobre los que no puedan insertarse subelementos, por lo que no se desplegará ninguna lista disponible. Encontraremos otros elementos obligatorios que LOMEditor no nos permitirá eliminar, y por tanto, como vemos por ejemplo en la captura anterior, no encontraremos esa opción. Finalmente, si el elemento no contiene ningún parámetro modificable, sencillamente estos no aparecerán.

En la siguiente captura, sin embargo, vemos desplegarse un mayor número de opciones. Se trata de un elemento de metadata.

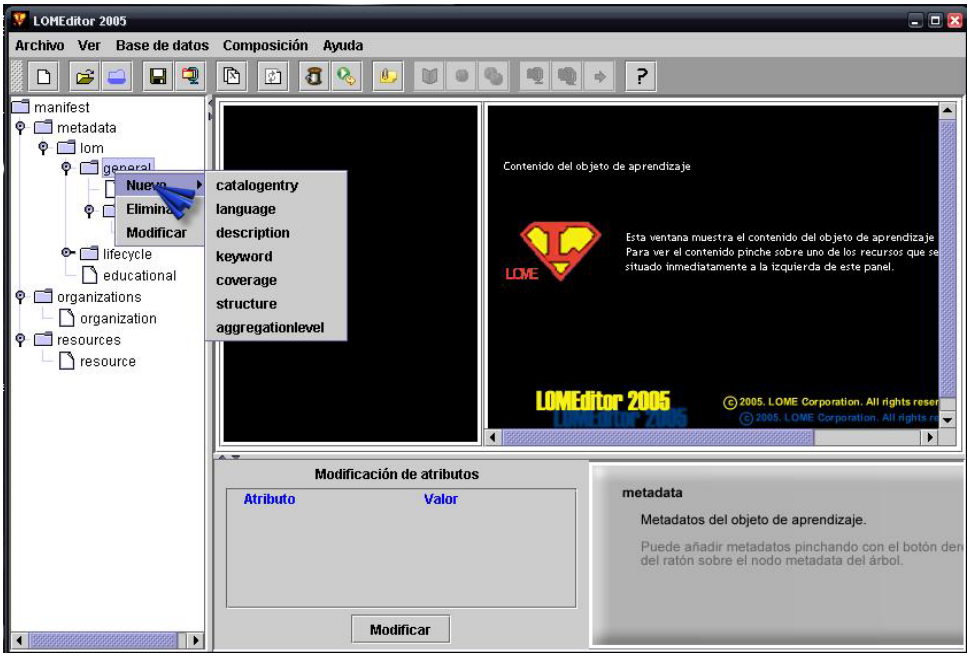


Imagen B 4.1.9 – Ejemplo edición metadata

Al insertar o eliminar un elemento no se producirán más cambios que los realmente ordenados por nosotros. En la modificación, sin embargo, necesitaremos algunos pasos más para completar la operación.

Una vez hayamos pulsado el botón de modificación sobre el nodo objetivo, veremos aparecer los parámetros modificables en el panel de modificación inferior. Si no apareciese ninguno simplemente sucedería que el elemento no los contiene.

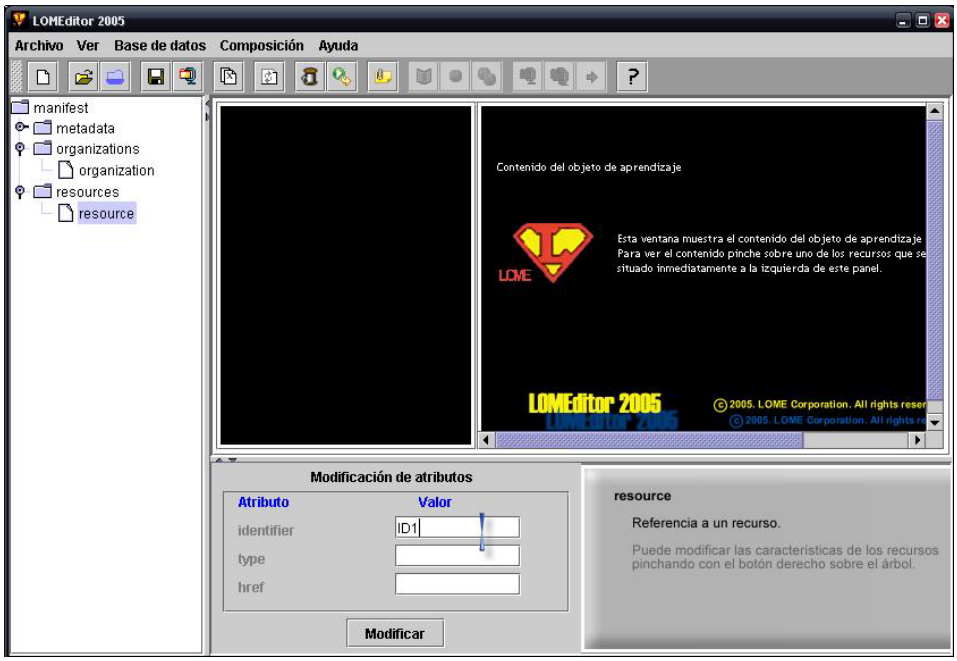


Imagen B 4.1.10 – Vista panel modificación

Cuando hayamos introducido el contenido adecuado en cada uno de ellos procederemos a guardar estas entradas en el elemento correspondiente; para ello basta con pulsar el botón de "Modificar" que encontraremos bajo el panel de modificación. Recordamos que cualquier modificación no guardada de esta forma se perderá.

5. Visualizar el contenido del objeto abierto

Cuando hayamos creado el objeto de aprendizaje, o habiendo abierto alguno ya completado, podremos ver su contenido. No hay que olvidar que un objeto de aprendizaje tiene entre sus objetivos el mostrado de recursos audiovisuales, documentos, etc.

La visualización del objeto es muy sencilla. Cuando hayamos incluido un recurso en el objeto y le hayamos asignado una organización, este aparecerá reflejado en el panel de visualización, como un enlace.

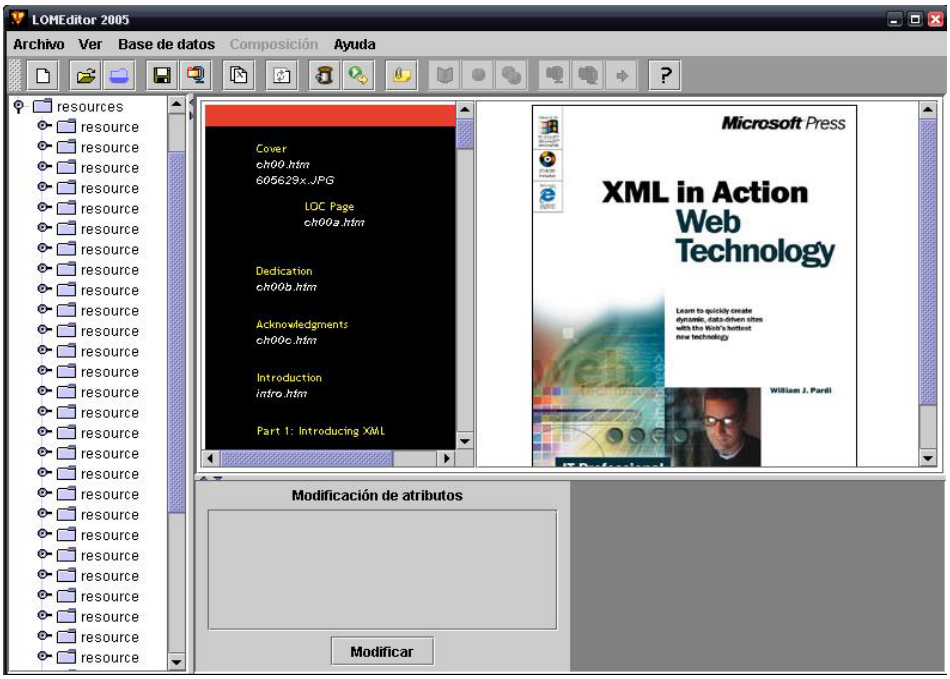


Imagen B 4.1.11 – Visionado de recursos

Pulsando sobre un enlace podremos ver el contenido del fichero en el panel derecho. Podemos además refrescar el mostrado, como se muestra en la siguiente captura.

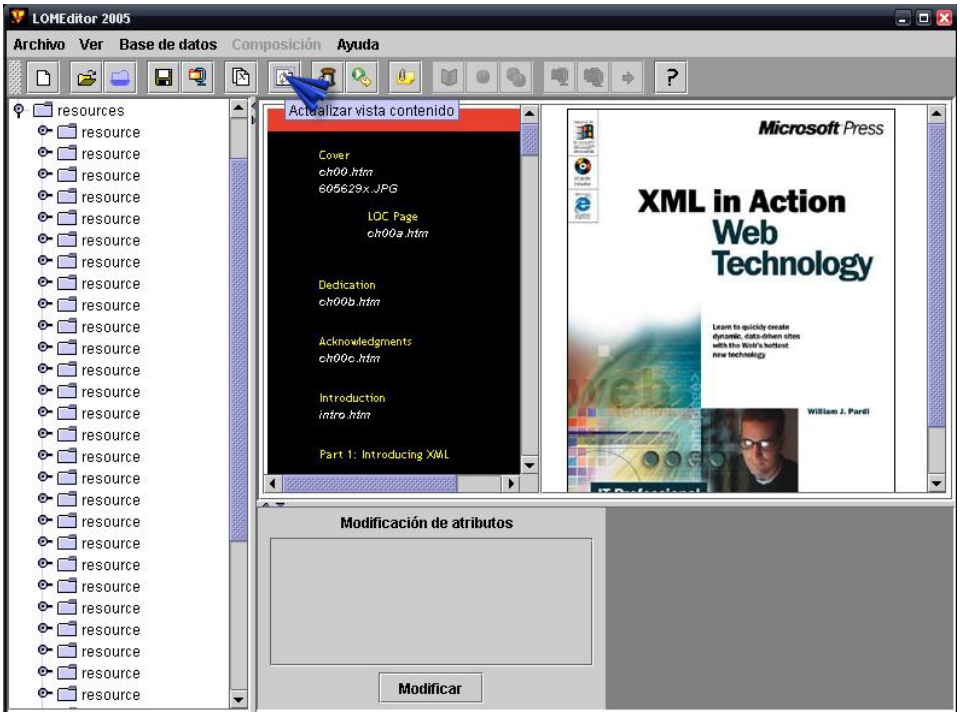


Imagen B 4.1.12 – Refrescar imágenes

En caso de tratarse de un fichero de tipo no estándar (ficheros .doc, .pdf, etc.) LOMEditor recurrirá a las aplicaciones dedicadas para su apertura, pidiendo antes nuestro consentimiento.

B.4.2 Composición

La composición de objetos de aprendizaje supone una innovación en este campo, que nos van a permitir generar objetos de forma más rápida y reutilizable.

Es necesario partir de dos o más objetos, como resulta lógico, a la hora de realizar un proceso de composición. Nosotros trabajaremos con los objetos de aprendizaje generados anteriormente, en busca de una mayor claridad.

Para aprovechar esta funcionalidad debemos seguir los pasos que se detallan a continuación:

En primer lugar accederemos al modo composición; para hacerlo basta con pulsar sobre el botón destinado a este fin.

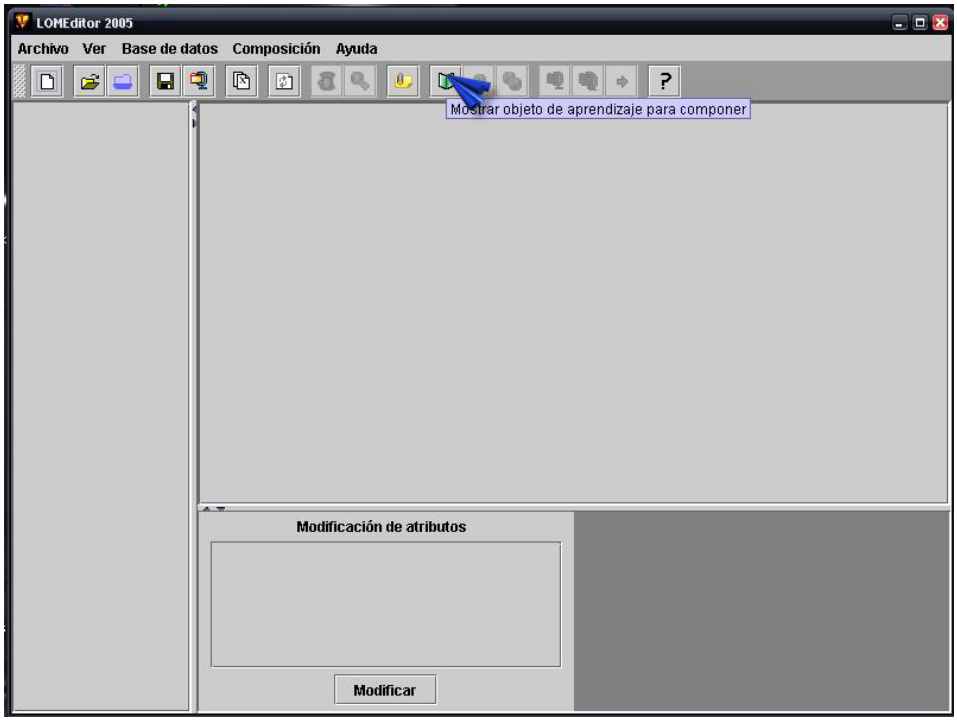


Imagen B 4.2.1 – Acceso composición

Lo primero que hará LOMEditor es pedirnos que seleccionemos, explorando el sistema mediante un cuadro de diálogo, el primer objeto para realizar la composición.

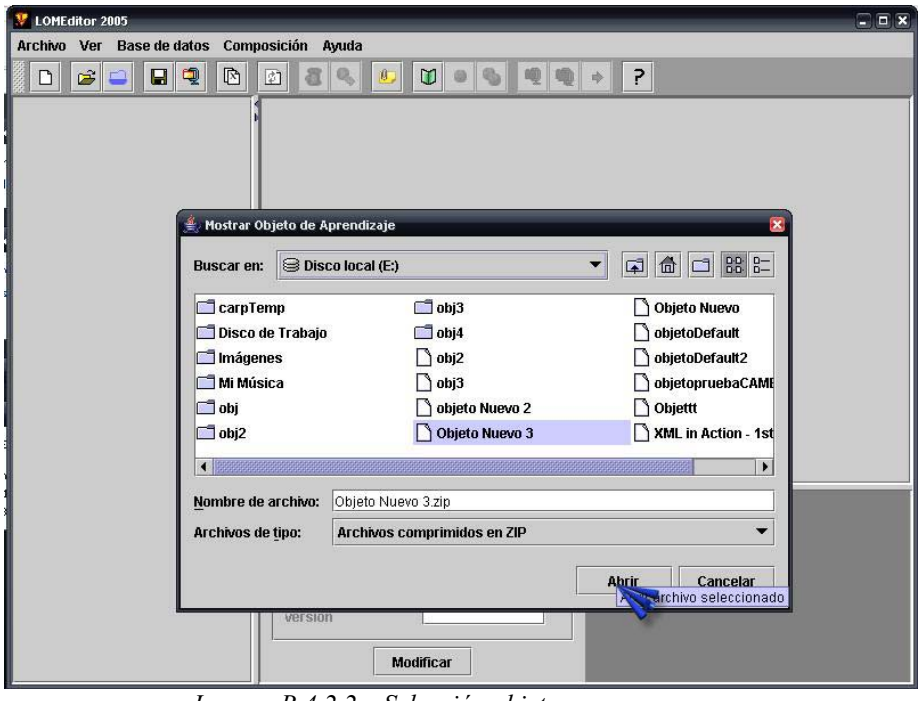


Imagen B 4.2.2 – Selección objeto para componer

El objeto quedará abierto, aunque sólo se nos mostrará el elemento organizations, pues es el relevante en este caso.

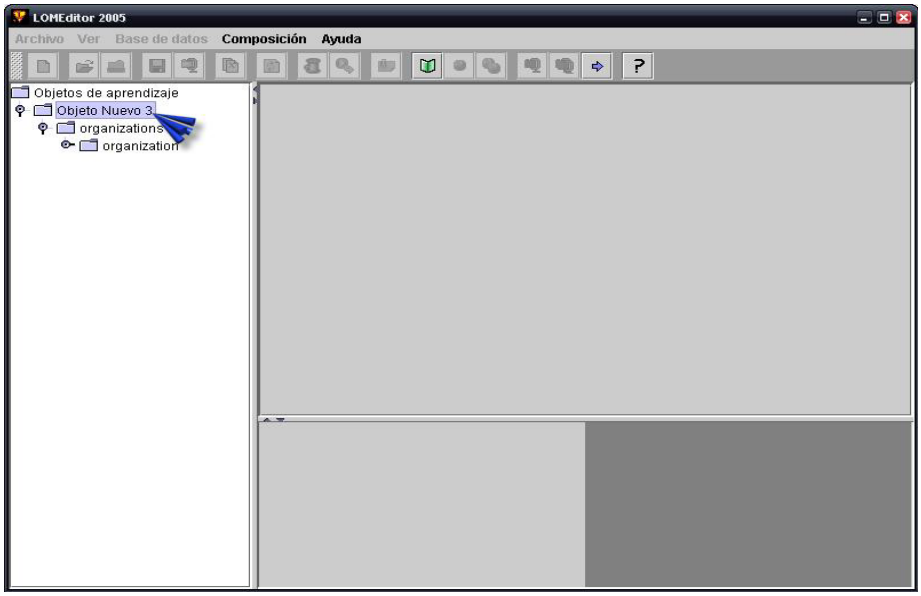


Imagen B 4.2.3 – Visionado objeto para composición

Debemos tener en cuenta que no se abrirán dos objetos con el mismo nombre o el mismo identificador. Además, los objetos seleccionados han de ser compatibles, es decir, no será posible componer dos objetos de gramáticas radicalmente diferentes.

Teniendo esto presente, buscamos el objeto que deseamos componer con el anteriormente abierto.

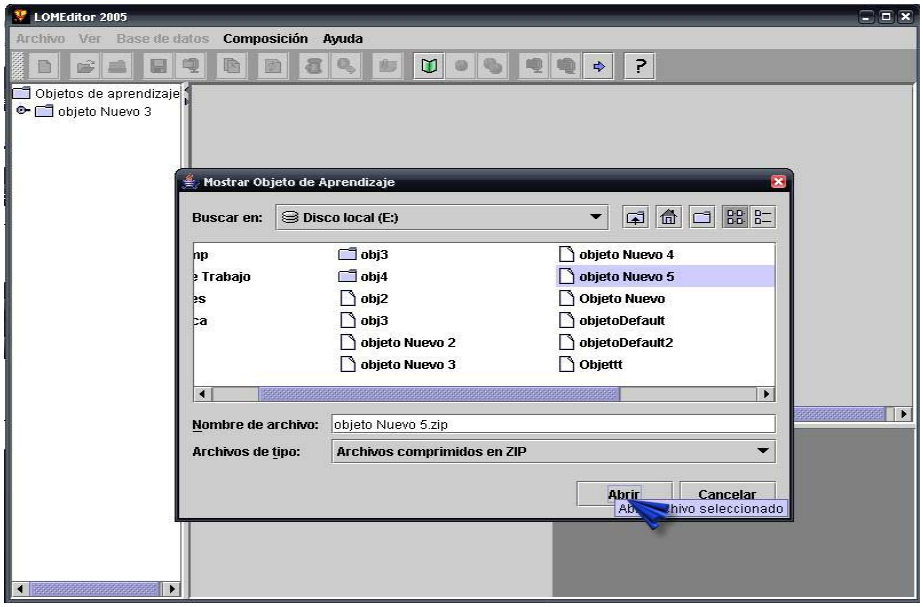


Imagen B 4.2.4 – Apertura segundo objeto composición

El objeto se situará junto al anterior, en el árbol de composición.

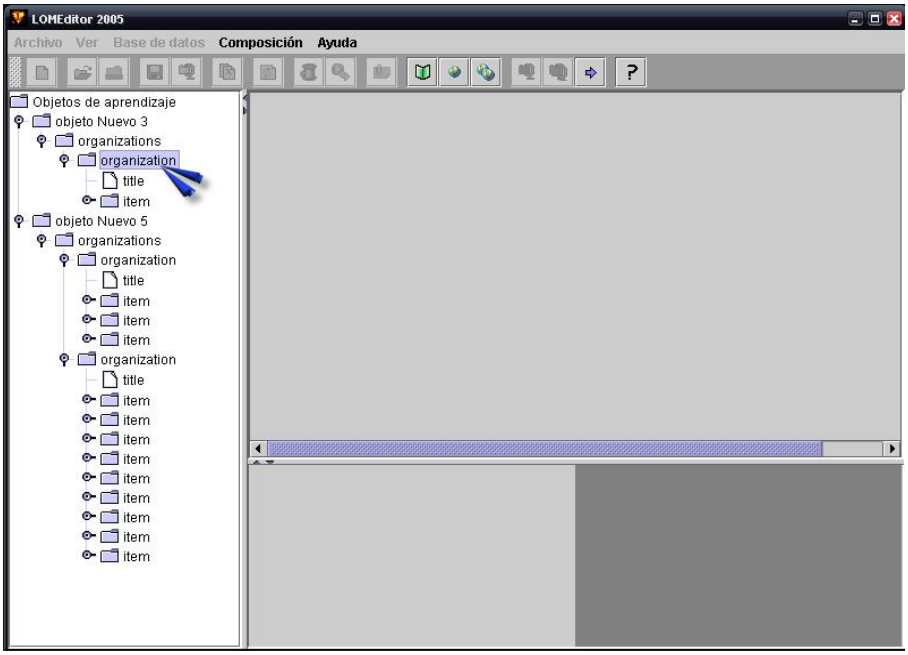


Imagen B 4.2.5 – Visión objetos en composición

Podemos mostrar en el árbol de composición tantos objetos de aprendizaje como deseemos para su posible composición, aunque la composición se realice únicamente entre dos de estos objetos.

En este tutorial mostraremos en el árbol únicamente dos objetos de aprendizaje, con el fin de no complicar excesivamente el proceso.

Nos encontramos ahora en el verdadero proceso de composición de objetos de aprendizaje; debemos decidir si vamos a realizar una composición en paralelo (donde al componer dos objetos, la organización elegida del objeto de aprendizaje que actúa como contenido, pasa a ser un hijo directo de la raíz de una organización del objeto que actúa como continente), o una composición en profundidad (en la cual el objeto de aprendizaje que actúa como contenido, va a ser un hijo de algún hijo interno de la raíz de una organización del objeto que actúa como continente).

La interfaz dispone de ambos servicios, accesibles tanto a nivel de barra de herramientas como de menú.

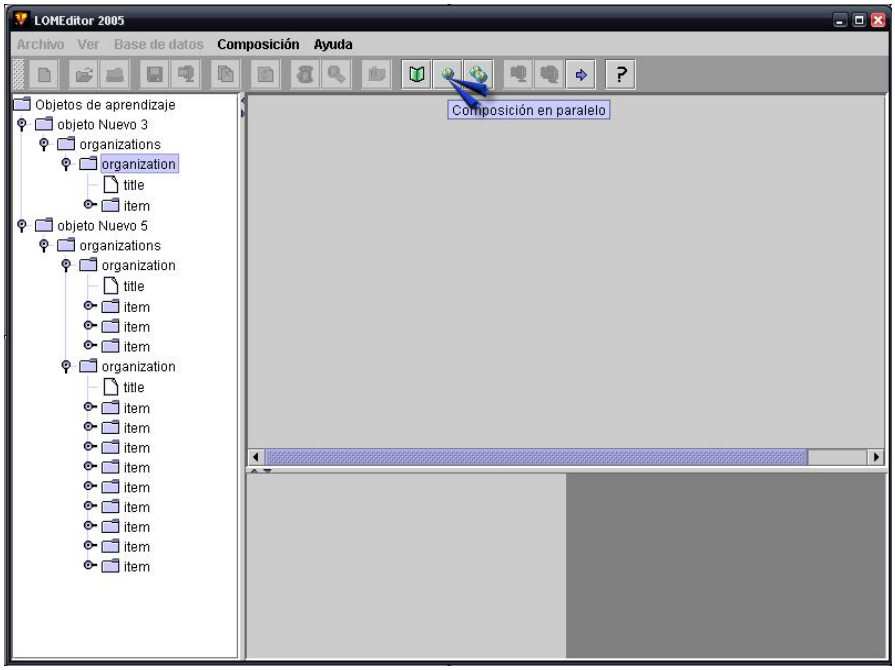


Imagen B 4.2.6 – Composición paralelo

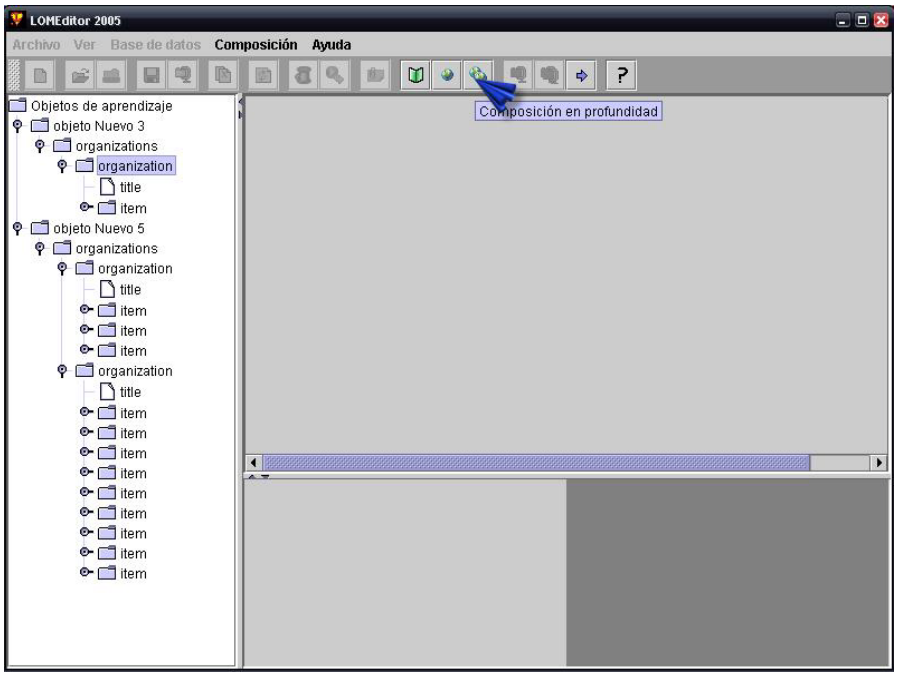


Imagen B 4.2.7 – Composición profundidad

Vamos a ver las diferencias entre cada estilo de composición, tanto paralela como en profundidad.

1. Composición en paralelo

Para esta composición seleccionaremos el primer botón mostrado en las capturas anteriores.

En ese momento el sistema realizará las peticiones necesarias para lograr realizar esta composición. Debemos seleccionar las organizaciones que entrarán en juego durante la composición, eligiendo de esta forma los objetos padre e hijo.

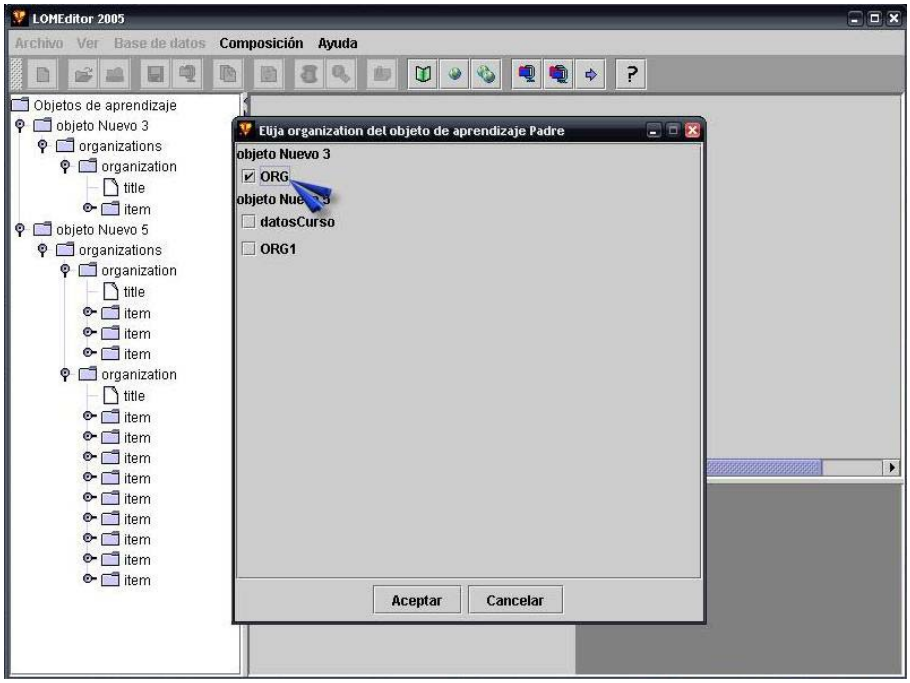


Imagen B 4.2.8– Elección padre paralelo

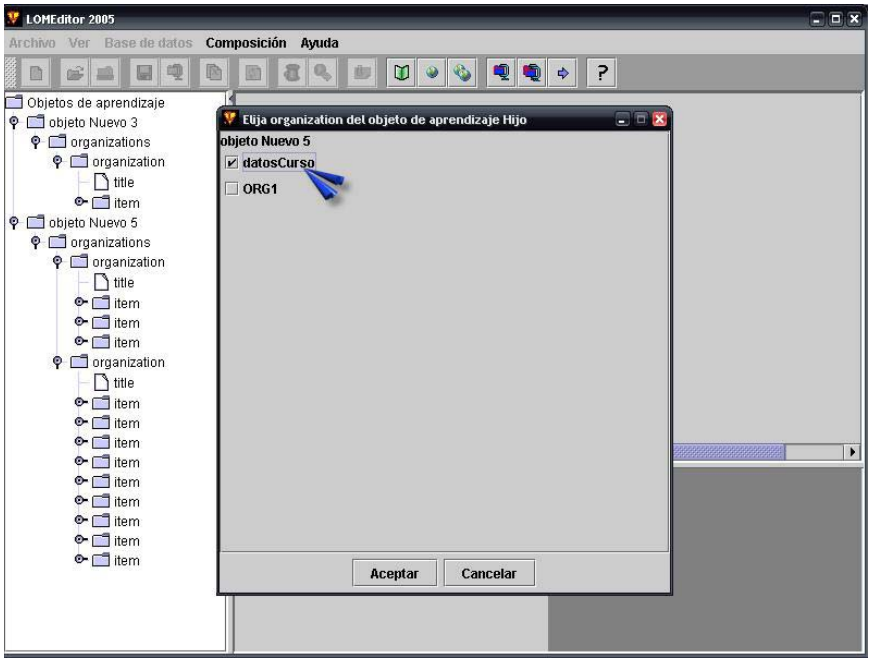


Imagen B 4.2.9 – Elección hijo paralelo

Cuando pulsemos *aceptar* en este formulario, la herramienta LOMEditor procederá a componer los objetos, mostrando el resultado como un objeto de aprendizaje disponible para nuevas composiciones.

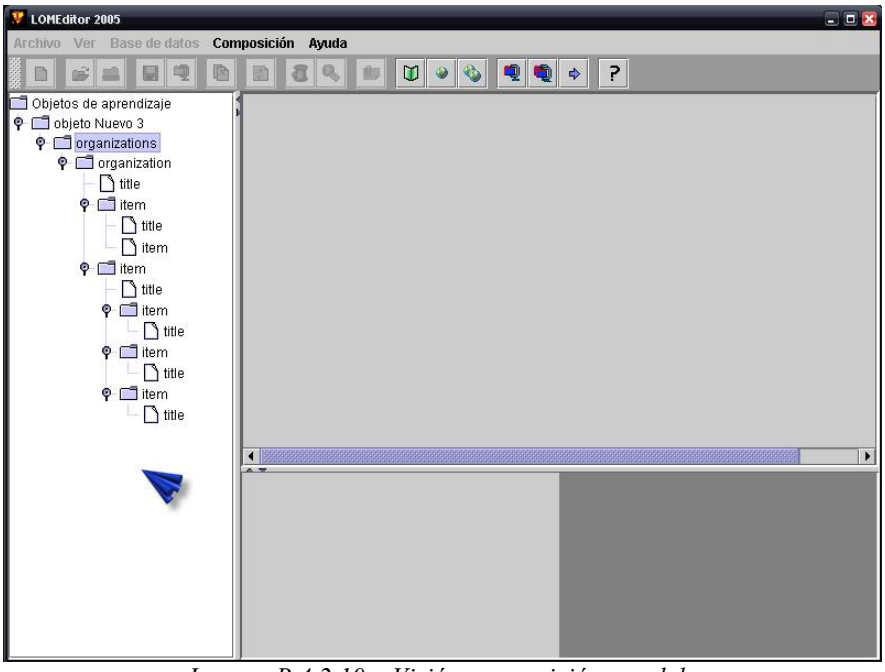


Imagen B 4.2.10 – Visión composición paralelo

El resultado es claro. El nuevo objeto de aprendizaje compuesto contiene como un ítem la organización original del hijo. Ahora este objeto compuesto puede ser guardado y utilizado de esta forma.

Todos los cambios en consecuencia son realizados automáticamente por LOMEditor, y puede observarse cuando el objeto compuesto sea guardado y nuevamente abierto para su edición o visualización.

2. *Composición en profundidad*

Vamos a ver también esta segunda modalidad de la composición. Su realización es similar a la anterior. Seleccionaremos el botón de Composición en profundidad, localizado en el menú o en la barra de herramientas.

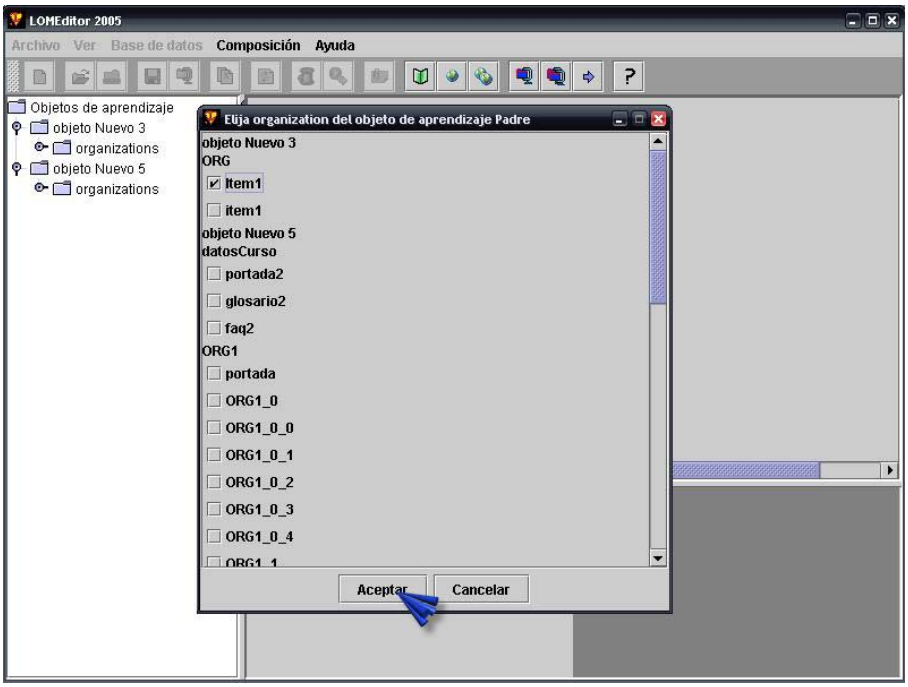


Imagen B 4.2.11 – Elección padre profundidad

Como vemos, aparecerá una ventana donde elegir el ítem del objeto que actuará como padre. Inmediatamente después seleccionaremos la organización hija.

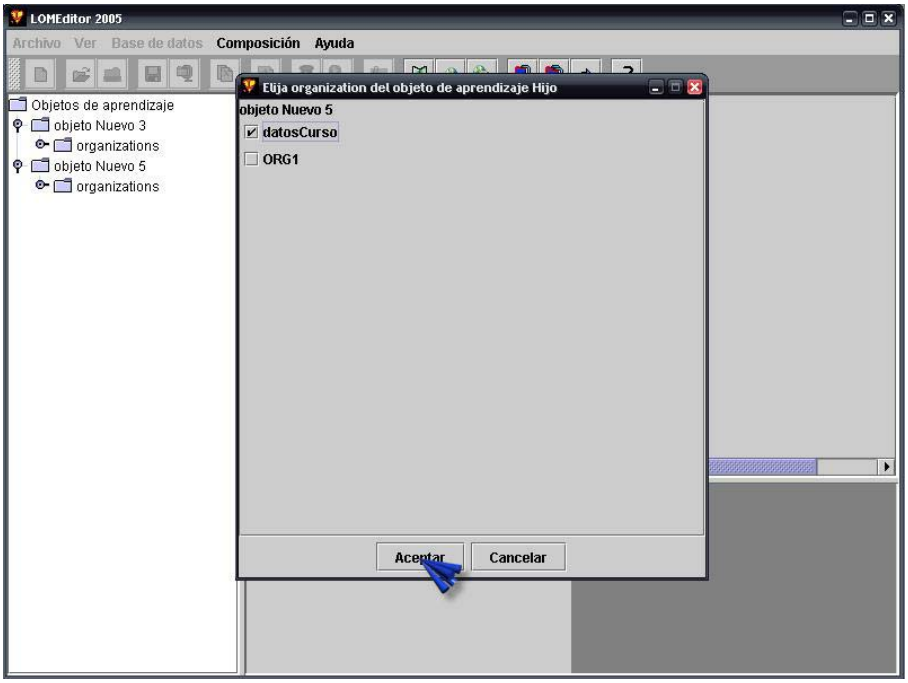


Imagen B 4.2.12 – Elección hijo profundidad

De nuevo volveremos a pulsar *Aceptar* para que LOMEditor pueda terminar la composición de los objetos de aprendizaje seleccionados. Esta vez, como veremos a continuación, la composición se hace a nivel jerárquico, lo cual ofrece la posibilidad de subdividir las organizaciones con las que estamos trabajando.

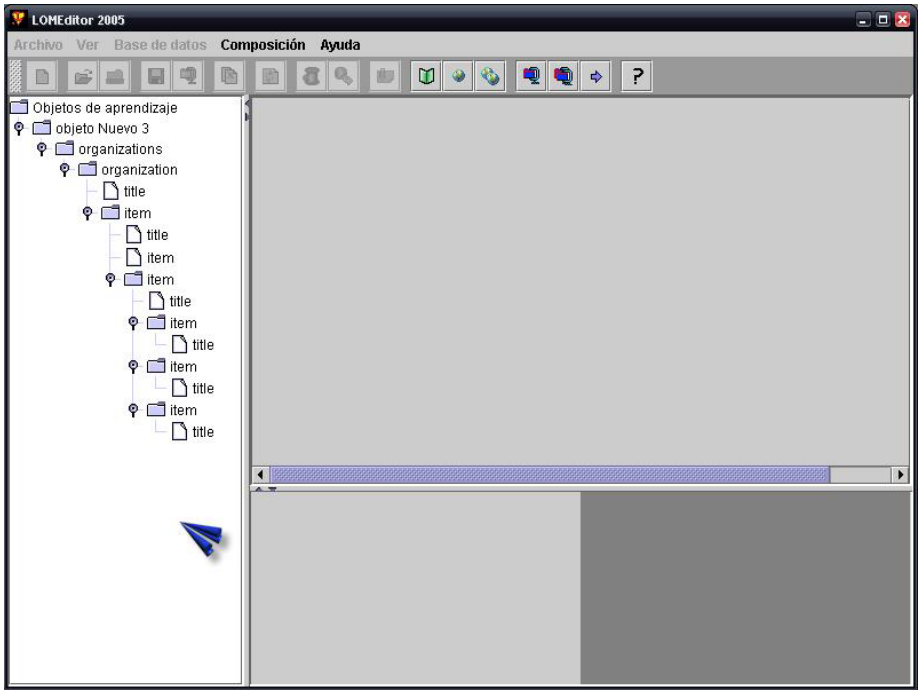


Imagen B 4.2.13 – Visión composición profundidad

Lo vemos más claro gráficamente, donde podemos observar la estructura escalonada que toma el contenido de organization, respecto a la anterior composición.

Hasta aquí hemos visto cómo realizar la composición entre varios objetos de aprendizaje anteriormente creados. Ahora, para finalizar este punto, y con ello finalizar la composición de nuestro nuevo objeto de aprendizaje, veremos cómo guardarlo y abandonar el módulo de composición.

Así como LOMEditor permite guardar los objetos de aprendizaje que se están editando en la herramienta, es posible salvar el objeto compuesto que hemos desarrollado. Prestemos atención, pues no utilizaremos los mismos botones que usamos para guardar anteriormente.

Dependiendo de si deseamos guardar un objeto determinado, o todos los objetos compuestos, seleccionaremos "guardar objeto compuesto como..." o "guardar todos los objetos compuestos".

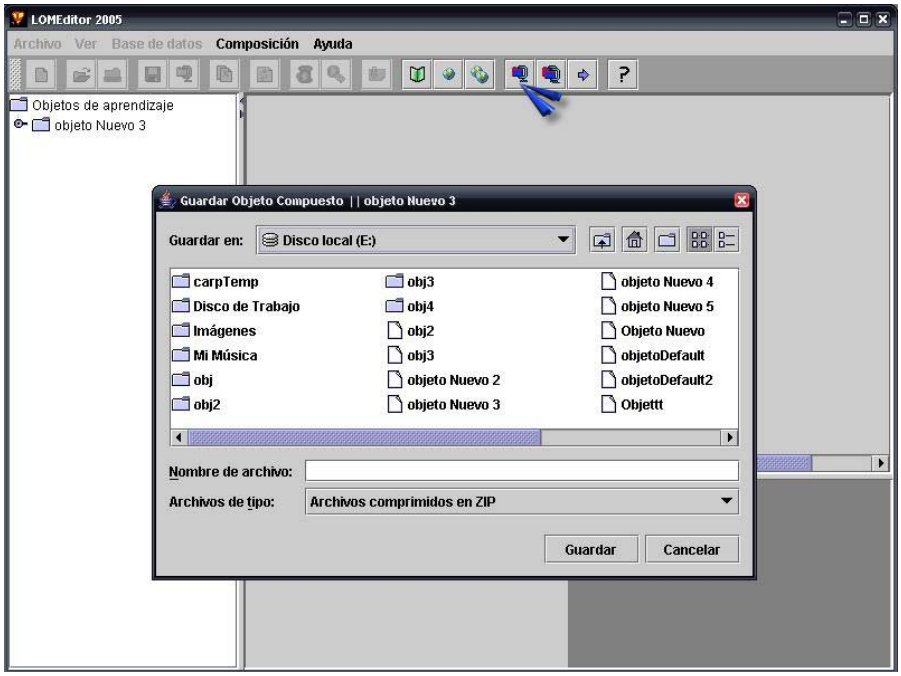


Imagen B 4.2.14 – Guardar objeto compuesto

Los pasos a seguir a partir de aquí en adelante son muy sencillos; indicaremos al sistema, mediante un cuadro de diálogo, dónde y con qué nombre guardar nuestro nuevo objeto generado por composición, y lo salvaremos.

Finalmente, para abandonar la composición disponemos de un nuevo botón destinado exclusivamente a este fin. Para abandonar la composición deberemos dar nuestro consentimiento a la herramienta, de la forma que se ve a continuación.

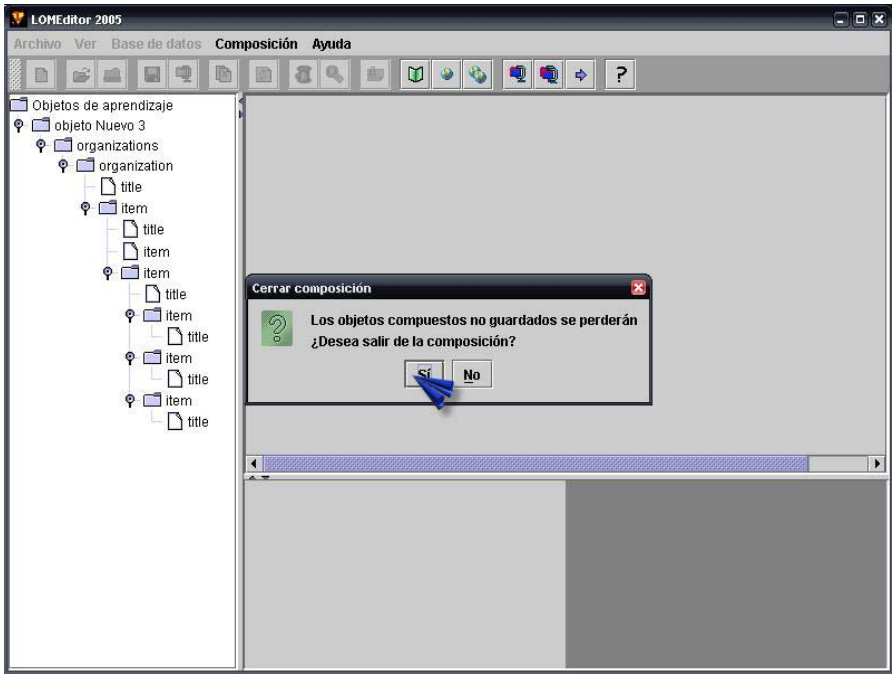


Imagen B 4.2.9 – Abandonar modo composición

Apareceremos de nuevo en la herramienta LOMEditor vacía, y lista para volver a trabajar sobre cualquier módulo del editor. Contaremos además con un nuevo objeto de aprendizaje, creado mediante métodos avanzados.

B.4.3 Evaluación de la calidad

Este último módulo tiene como finalidad prestarnos un soporte en la edición de objetos de aprendizaje, aportándonos una evaluación de la calidad del objeto con el que estamos tratando. Nos será de gran utilidad a la hora de contrastar diferentes objetos, o plantearnos objetivos.

No entraremos en complejos detalles de implementación, sino que nos limitaremos a realizar un paseo por esta innovadora funcionalidad. Decir únicamente que, dado que el razonamiento de estas evaluaciones se basa en casos específicos, habrá que colaborar con el sistema presentándole los mismos, ya sea para evaluarlos nosotros manualmente, o permitir que el mismo LOMEditor le asigne una evaluación.

Un detalle importante a tener en cuenta es que para lograr evaluar un objeto de aprendizaje, este debe contar con una descripción técnica del mismo, es decir, necesita contener metadatos.

Antes de nada, por supuesto, debemos haber abierto o tener actualmente algún objeto de aprendizaje en la herramienta.

1. Evaluación manual

En esta modalidad seremos nosotros mismos los que, de acuerdo a unas puntuaciones, asignemos la calidad que creemos oportuna al objeto. Como anteriormente hemos hecho mención, serán los elementos de Metadata los llevados a examen. Cuanto más completa y detallada sea esta descripción del objeto, mejor puntuación obtendrá.

Para acceder a la evaluación manual, debemos dirigirnos al botón destinado a ello, como vemos en la imagen.

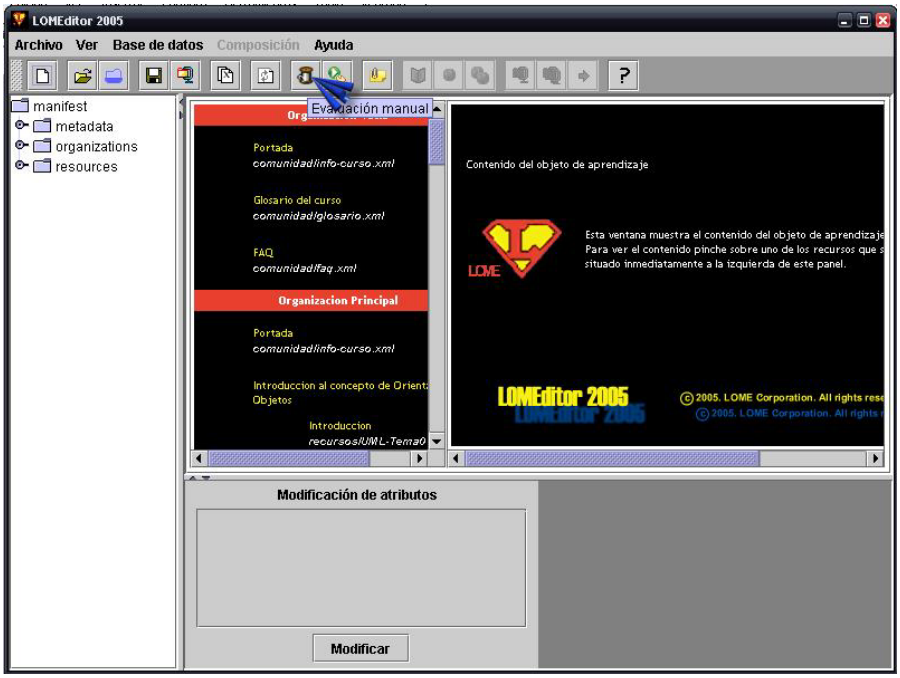


Imagen B 4.3.1 – Selección de evaluación manual

Nos será mostrada una ventana donde encontraremos listados los diferentes parámetros analizables. Una vez les hayamos dado peso, podremos introducir esta evaluación en el sistema.

Imagen B 4.3.2 –Evaluación manual

Es conveniente realizar este paso en varias ocasiones con diferentes objetos, pues de esta manera no solo reforzaremos el conocimiento de LOMEditor, sino que además fijaremos unos referentes a la hora de puntuar los objetos.

2. *Evaluación automática*

En esta ocasión, gracias al conocimiento adquirido por la herramienta, y a su sistema de razonamiento, será el mismo LOMEditor el que asigne la puntuación de calidad adecuada al objeto.

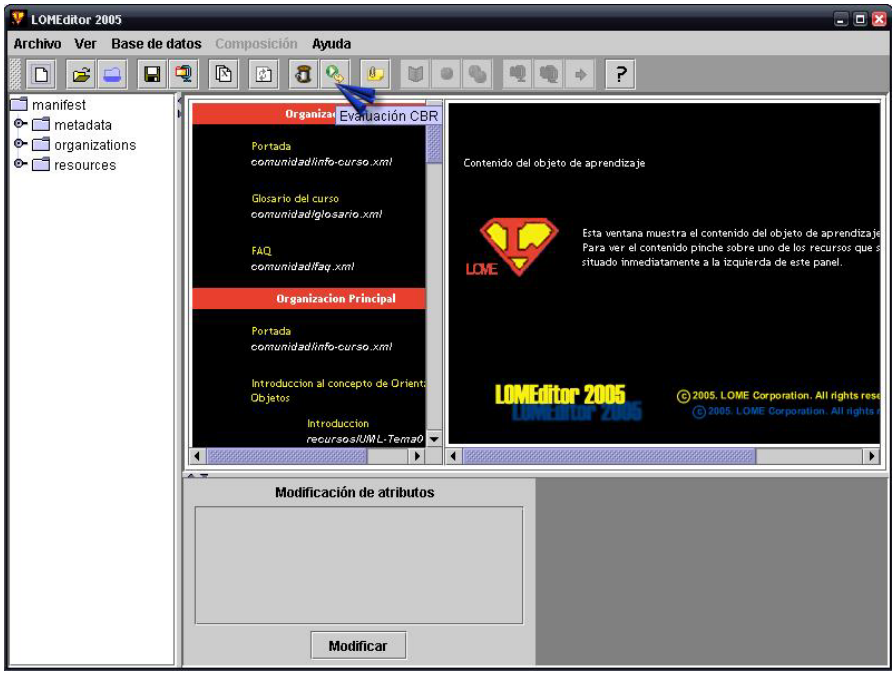


Imagen B 4.3.3 – Selección de evaluación automática

Vemos cómo muestra la puntuación en el siguiente panel.

Imagen B 4.3.4 – Evaluación general

Por supuesto, existe la posibilidad de modificar esta evaluación en caso de no estar satisfechos. De esta manera LOMEditor no tendrá en cuenta su propio cálculo, sino que fijará el que le hemos determinado.

3. Consulta a la Base de Datos

LOMEditor almacena los resultados en la base de datos, junto con un esquema del objeto al que pertenecen. Gracias al avanzado sistema de consulta, es posible recuperar los datos del objeto mediante este servicio.

Accederemos al botón que envía la petición al sistema, el cual nos mostrará una ventana donde rellenar el formulario de consulta. Cuanto más estrictos seamos, menos objetos recuperaremos. A veces conviene ser un poco flexible en este punto, y realizar más tarde la selección definitiva. Dejamos, no obstante, en manos del usuario esta decisión.

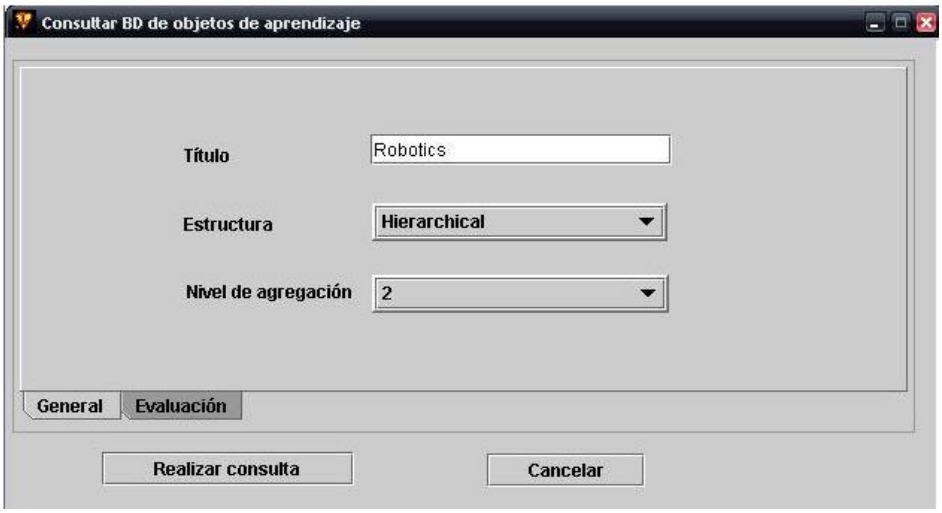


Imagen B 4.3.5 – Consulta a la base de datos

Al realizar la consulta veremos resultados de esta forma siguiente.

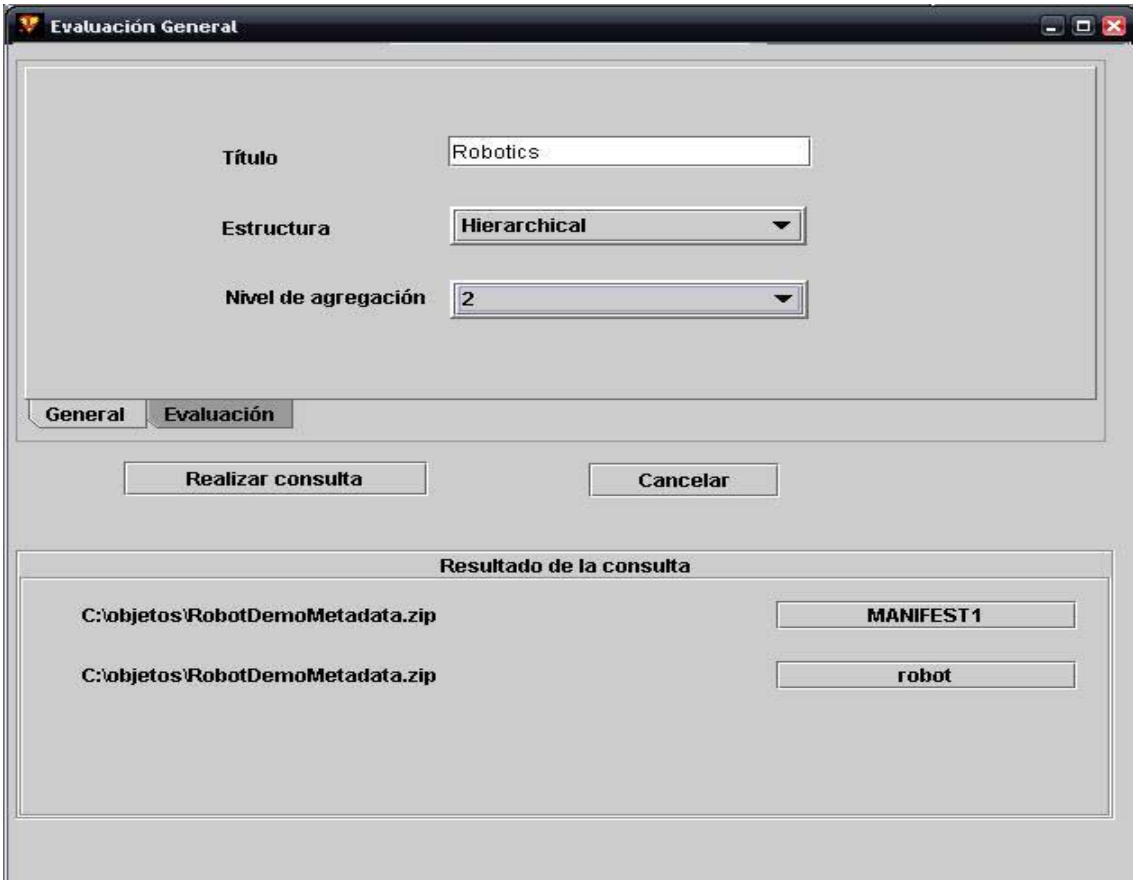


Imagen B 4.3.6 – Resultado consulta a la base de datos

De esta forma LOMEditor permite a los usuarios realizar evaluaciones de calidad, y consultar, de acuerdo a unos criterios, la base de datos generada durante estos procesos.

APÉNDICE C

FICHEROS DE CONFIGURACIÓN

C.1 Ficheros relacionados con la visualización de contenidos

C.1.1 *objetoAprendizaje.xsl*

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:ra="http://www.imsglobal.org/xsd/imscp_v1p1"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:fo="http://www.w3.org/1999/XSL/Format"
xmlns:fn="http://www.w3.org/2005/02/xpath-functions">
   <xsl:output method="html"/>
   <!--
**********************************************************************************************************************
*** -->
   <xsl:template match="/">
        <html>
            <head>
                <title>
                    Contenido del objeto de aprendizaje
                </title>
                <xsl:element name="link">
                    <xsl:attribute name="rel">stylesheet</xsl:attribute>
                    <xsl:attribute name="type">text/css</xsl:attribute>
                    <xsl:attribute name="href">objAprendizaje.css</xsl:attribute>
                </xsl:element>
            </head>
            <body>
                <xsl:apply-templates/>
            </body>
        </html>
   </xsl:template>
   <!--
**********************************************************************************************************************
*** -->
   <xsl:template match="ra:organizations" name="organizations">
        <xsl:apply-templates/>
   </xsl:template>
   <!--
**********************************************************************************************************************
*** -->
   <xsl:template match="ra:organization" name="organization">
        <table>
            <thead>
                <tr>
                    <th><xsl:value-of select="ra:title"/></th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td><xsl:apply-templates/></td>
                </tr>
            </tbody>
        </table>
   </xsl:template>
   <!--
**********************************************************************************************************************
*** -->
   <xsl:template match="ra:item" name="item">
        <ul>
            <xsl:value-of select="ra:title"></xsl:value-of>
            <xsl:call-template name="resources">
                <xsl:with-param name="id" select="@identifierref"></xsl:with-param>
            </xsl:call-template>
            <xsl:apply-templates/>
```

```
        </ul>
    </xsl:template>
    <!--
**********************************************************************************************************************
*** -->
    <xsl:template match="ra:metadata">
    </xsl:template>
    <!--
**********************************************************************************************************************
*** -->
    <xsl:template name="resources">
        <xsl:param name="id"/>
        <xsl:for-each select="//ra:manifest/ra:resources/ra:resource">
            <xsl:if test="$id=@identifier">
                <xsl:for-each select="ra:file">
                    <li>
                        <xsl:element name="a">
                            <xsl:attribute name="target">mainFrame</xsl:attribute>
                            <xsl:attribute name="href"><xsl:value-of select="@href"/></xsl:attribute>
                            <xsl:value-of select="@href"/>
                        </xsl:element>
                    </li>
                </xsl:for-each>
            </xsl:if>
        </xsl:for-each>
    </xsl:template>
    <!--
**********************************************************************************************************************
*** -->
    <xsl:template match="ra:resources">
    </xsl:template>
    <!--
**********************************************************************************************************************
*** -->
    <xsl:template match="ra:title">
    </xsl:template>
    <!--
**********************************************************************************************************************
******-->
    <xsl:template match="ra:manifest">
        <xsl:apply-templates/>
    </xsl:template>
    <!--
**********************************************************************************************************************
******-->
</xsl:stylesheet>
```

C.1.2 *objetoAprendizaje.css*

```
body
{
	background-color: #000000;
 }
table
{
	width: 200px;
	color: #FFFFFF;
	background-color: #E7402E;
	font-family: "Lucida Grande", "Trebuchet MS", Verdana, Arial, sans-serif;
	font-size: 8px;
}

td {
	background-color: #000000;
	color:#F9F112;
}

table.sencilla
{
	color: #FFFFFF;
	background-color: #000000;
	width: 500px;
	font-family: "Lucida Grande", "Trebuchet MS", Verdana, Arial, sans-serif;
	font-size: 8px;
}


td.sencilla
{
	background-color: #000000;
	color:#FFFFFF;
}


a {
	text-decoration: none;
	font-weight: bold;
	color: #FFFFFF;
	border-bottom: none;
	font-family: Arial, "Trebuchet MS", Verdana, sans-serif;
	font-size: 8px;
	font-style: italic;
}
a:hover {
	font-family: Arial, "Trebuchet MS", Verdana, sans-serif;
	font-size: 8px;
	font-style: italic;
	border-bottom: dotted 1px #FFFFFF;
}

li
{
	list-style-image: none;
	list-style-position:outside;
}

ul
{
	margin-left:20;
}
```

C.1.3 *viewFrames.html*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
"http://www.w3.org/TR/html4/frameset.dtd">
<html>
<head>
<title>Documento sin t&iacute;tulo</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>

<frameset cols="230,*" frameborder="NO" border="0" framespacing="0">
  <frame src="view.html" name="leftFrame">
  <frame src="lome.html" name="mainFrame">
</frameset>
<noframes><body>
</body>
</noframes>
</html>
```

C.1.4 *lome.html*

```
<html>
   <head>
   <title>Contenido del objeto de aprendizaje</title>
   <link rel="stylesheet" type="text/css" href="objAprendizaje.css">
   </head>
   <body>
   <table width="524" height="279"  border="0" class="sencilla">
     <tr>
       <td width="1" class="sencilla"> </td>
       <td height="92" colspan="3" class="sencilla"><p>Contenido del objeto de aprendizaje</p>    </td>
     </tr>
     <tr>
       <td class="sencilla"> </td>
       <td width="84" height="59" class="sencilla"><img src="superlomeNegro.png" width="99"
   height="70"></td>
       <td width="396" align="left" valign="top" class="sencilla"><p>Esta ventana muestra el contenido del
   objeto de aprendizaje abierto actualmente. <br>
       Para ver el contenido pinche sobre uno de los recursos que se muestran en el men&uacute; situado
   inmediatamente a la izquierda de este panel. </p>    </td>
       <td width="1" class="sencilla"> </td>
     </tr>
     <tr>
       <td height="65" colspan="4" class="sencilla"> </td>
     </tr>
     <tr>
       <td height="53" colspan="4" class="sencilla"><img src="lomeditor2.png" width="450"
   height="51"></td>
     </tr>
   </table>
   </body>
</html>
```

C.1.5 *view.html.* Ejemplo generado a partir de la aplicación de objetoAprendizaje.xsl al manifiesto de un objeto de aprendizaje.

```
<html xmlns:fn="http://www.w3.org/2005/02/xpath-functions"
xmlns:fo="http://www.w3.org/1999/XSL/Format" xmlns:ra="http://www.imsglobal.org/xsd/imscp_v1p1">
<head>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>
                        Contenido del objeto de aprendizaje
                    </title>
<link rel="stylesheet" type="text/css" href="objAprendizaje.css">
</head>
<body>

<table>
<thead>
<tr>
<th>xR 500 Robotic Arm</th>
</tr>
</thead>
<tbody>

<tr>
<td>Overview<tr>
<td><a target="marcoMostrar.html"
href="robot2/background/robotintro.htm">robot2/background/robotintro.htm</a></td>
</tr>

<tr>
<td>xR 500 Mechanics<tr>
<td><a target="marcoMostrar.html" href="robot2/intro2.htm">robot2/intro2.htm</a></td>
</tr>
<tr>
<td><a target="marcoMostrar.html" href="robot2/intro.dcr">robot2/intro.dcr</a></td>
</tr>
<tr>
<td><a target="marcoMostrar.html" href="robot2/rob1.dcr">robot2/rob1.dcr</a></td>
</tr>

</td>
</tr>

</td>
</tr>

<tr>
<td>Procedure<tr>
<td><a target="marcoMostrar.html"
href="robot2/background/robpro.htm">robot2/background/robpro.htm</a></td>
</tr>

<tr>
<td>Introduction<tr>
<td><a target="mainFrame" href="robot2/rob2.dcr">robot2/rob2.dcr</a></td>
</tr>

</td>
</tr>

<tr>
<td>Procedure<tr>
<td><a target="marcoMostrar.html" href="robot2/rob3.dcr">robot2/rob3.dcr</a></td>
</tr>
```

```
</td>
</tr>

<tr>
<td>Conclusion<tr>
<td><a target="marcoMostrar.html" href="robot2/rob4.dcr">robot2/rob4.dcr</a></td>
</tr>

</td>
</tr>

</td>
</tr>

<tr>
<td>Simulation<tr>
<td><a target="marcoMostrar.html"
href="robot2/background/robsim.htm">robot2/background/robsim.htm</a></td>
</tr>

<tr>
<td>Introduction<tr>
<td><a target="marcoMostrar.html" href="robot2/rob45.dcr">robot2/rob45.dcr</a></td>
</tr>

</td>
</tr>

<tr>
<td>Simulation of Procedure<tr>
<td><a target="marcoMostrar.html" href="robot2/rob5.dcr">robot2/rob5.dcr</a></td>
</tr>

</td>
</tr>

</td>
</tr>

<tr>
<td>Credits<tr>
<td><a target="marcoMostrar.html" href="robot2/end.dcr">robot2/end.dcr</a></td>
</tr>
</td>
</tr>
</tbody>
</table>

</body>
</html>
```

C.2 Fichero plantilla para objeto nuevo

C.1.2.1 *imsmanifest.xml*

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<manifest xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_v1p2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" identifier="MANIFEST-1"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1 imscp_v1p1.xsd
http://www.imsglobal.org/xsd/imsmd_v1p2 imsmd_v1p2p2.xsd">
   <metadata>
        <imsmd:lom>
   </imsmd:lom>
   </metadata>
   <organizations default="ORG">
  </organizations>
   <resources>
  </resources>
</manifest>
```

APÉNDICE D

GLOSARIO

- *Algoritmo de validación:* Algoritmo que comprueba la corrección del fichero *imsmanifest.xml*, basándose en las gramáticas que definen su estructura, *xsd* de *Content Packaging* y de *Metadata*.

- *Algoritmo de recuperación:* Algoritmo que se encarga de recuperar el caso de la base de casos de un sistema CBR más similar al que se está evaluando.

- *Árbol de composición:* Árbol que muestra todos los objetos de composición que han sido abiertos para su posible composición.

- *Atributo:* En XML, definición de tipo simple con nombre que no puede contener otros elementos.

- *Base de casos:* Soporte físico que almacena el conocimiento de la aplicación en los sistemas CBR.

- *Caso:* Fragmento contextualizado de conocimiento que representa una experiencia y que enseña una lección importante para conseguir los objetivos del razonador. *[Kolodner & Leake 97]*

    - *Caso de prueba:* Caso recuperado de la base de casos del sistema.

- *CBR (Case Based Reasoning):* Tecnología para la construcción de sistemas expertos que basa su funcionamiento en un conjunto de problemas resueltos (base de casos) + conocimiento de similitud + conocimiento de adaptación.

- *Controlador:* Nexo de unión entre la Vista y el Modelo en el patrón Modelo Vista Controlador (MVC).

- *Composición en paralelo:* Una de las dos posibilidades de la composición, en la que el objeto de aprendizaje que actúa como contenido, será un hijo directo de la raíz de una organización del objeto que actúa como continente.

- *Composición en profundidad:* Uno de los dos tipos de composición en la que el objeto de aprendizaje que actúa como contenido, va a ser un hijo de algún hijo interno de la raíz de una organización del objeto que actúa como continente.

- *E-learning:* Iniciativa de adaptación de los elementos tradicionales de la enseñanza presencial al ámbito de los computadores, usando Internet y las nuevas tecnologías.

- *Elemento:* En XML etiqueta que puede contener: elementos anidados o hijos y atributos.

- *Gramática:* Ente que define la estructura de un documento. En este documento, cuando se hace referencia una gramática, se hace referencia a los ficheros xsd que definen organización del contenido de un objeto de aprendizaje.

- *Ítem:* Según *IMS Content Package*, etiqueta que representa un elemento en la estructura del manifiesto XML.

- *Lenguajes de marcado:* Los lenguajes de marcado son una manera de hacer explícita la estructura de cierto tipo de información sin especificar su presentación.

- *Manifiesto: (1)* Según *IMS Content Package*, etiqueta raíz del fichero que indica la estructura del objeto de aprendizaje.
  (2) El propio, fichero donde se define la estructura del objeto se conoce también como manifiesto.

- *Mapeo o parsing del manifiesto:* Proceso mediante el cual se almacena la estructura del fichero *imsmanifest.xml* en clases del sistema implementadas a tales efectos.

- *Medida de similitud:* Cómo se mide la relevancia de un caso para el problema a resolver.

- *Metadatos:* Según *IMS Content Package*, rama del árbol XML del manifiesto en la que se define la meta información asociada al objeto de aprendizaje.

- *Modelo:* Contenedor de los datos y la funcionalidad de la aplicación.

- *MVC:* Patrón Modelo Vista Controlador.

- *Objeto de aprendizaje:* Ente que encapsula información sobre contenidos didácticos, basándose en una especificación de contenidos.

- *Objeto de composición:* Objeto utilizado para la composición de objetos de aprendizaje, formado por el elemento Organizations del objeto de aprendizaje al que representa.

- *Objeto hijo o contenido:* En la composición es el objeto cuyos datos serán encapsulados en el primer objeto (objeto padre) que interviene en dicha composición.

- *Objeto padre o continente:* En la composición es el objeto que encapsulará los datos del segundo objeto que interviene en dicha composición (del objeto hijo).

- *Organización:* Según *IMS Content Package*, rama del árbol XML del manifiesto en la que se define la distribución de los elementos que compondrán el objeto de aprendizaje.

- *Recurso:* Según *IMS Content Package*, rama del árbol XML del manifiesto en la que se definen los recursos que contiene el objeto de aprendizaje.

- *Sub-manifiesto:* Según *IMS Content Package*, manifiesto que representa la estructura de un objeto de aprendizaje que se encuentra contenido a su vez en un objeto de aprendizaje padre o continente.

- *Vista:* Interfaz de interacción con el usuario en el Modelo Vista Controlador.

- *Zipear:* Compresión de archivos en formato zip.

APÉNDICE E

BIBLIOGRAFÍA

E.1 Referencias usadas

Entre las referencias usadas se pueden distinguir varios grupos:

E.1.1 Especificaciones o recomendaciones

[1]Advanced Distributed Learning Initiative. Sharable Courseware Object Reference Model (SCORM).

[2]IEEE Learning Technology Standards Committee (IEEE LTSC) Learning Object Metadata(LOM)

[3]IMS Global Learning Consortium.

[4] R. Koper, "Educational Modelling Language: adding instructional Design ton existing specifications".http://www.rz.uni-frankfurt.de/neue_medien/

[5]IEEE 1061-1998 Standard for a Software Quality Metrics Methodology

[6]Thierry Nodenot, Pierre Laforcade, Christophe Marquesuzaa, Christian Sallaberry." Knowledge Modelling of Co-operative Learning Situations: Towards a UML profile". 11th International Conference on Artificial Intelligence in Education.

E.1.2 Artículos sobre objetos de aprendizajes

[7]Douglas, I."Instructional design based on reusable learning objects: applying lessons of object-oriented software engineering to learning systems design". 31st Annual Frontiers in Education Conference, 2001.

[8] S. Downes , "Learning Objects"
http://www.atl.ualberta.ca/downes/naWeb/Learning_Objects.doc

[9]Friesen, N. (2003)." Three Objections to Learning Objects." Learning Objects and Metadata. (McGreal, R. ed.) London: Kogan Page.

[10] Cm Multimedia Systems."Learning Objects Does Size Matter?". Cm316 Multimedia Systems Coursework.

E.1.3 Artículos sobre autoría de recursos educativos

[11]Shih, T.K.; Lin, N.H.; Hsuam-Pu Chang . "An Intelligent E-Learning System with Authoring and Assessment Mechanism". Proceedings of the 17 th International Conference on Advanced Information Networking and Applications. AINA 2003.

[12] Mahadevan, S.; Rahman, S.; Wiesner, P. "An online process for posting and meta-tagging learning objects and integrating contents into digital libraries ".FIE 2002. 32nd Annual Frontiers in Education, 2002.

[13] Knolmayer, G.F. "Decision support models for composing and navigating through e-learning objects". Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003.

[14]SantaCruz-Valencia. L, Aedo. I, Breuer. T, Delgado Kloos C." A Framework for Creation, Integration and Reuse of Learning Objects". Learning Technology newsletter. Vol 5. Issue 1.

[15] Quarati, A. "Designing shareable and personalisable e-learning paths". Proceedings. ITCC 2003. International Conference on Information Technology: Coding and Computing [Computers and Communications], 2003.

[16] Kawai, H.; Takayama, F.; Anzai, T.; Manome, T.; Yoshida, H. "Objectives and features of e-learning oriented programming courseware for freshmen". Proceedings. 23rd International Conference on Distributed Computing Systems Workshops, 2003.

[17]Juan Manuel Dodero Beardo. "Una arquitectura multiagente para la producción distribuida de conocimiento y su aplicación al desarrollo compartido de objetos educativos" (2002). Tesis doctoral

E.1.4 Artículos sobre distribución de recursos educativos

[18] Shih, T.K.; Wen-Chih Chang; Lin, N.H.; Lin, L.H.; Hun-Hui Hsu; Ching-Tang Hsieh. "Using SOAP and .NET web service to build SCORM RTE and LMS". Proceedings of the 17 th International Conference on Advanced Information Networking and Applications. AINA 2003.

[19] Anido, L.; Llamas, M.; Fernandez, M.J.; Rodriguez, J.; Santos, J.; Caeiro, M. "A distributed object computing approach to e-learning" Proceedings. 3rd International Symposium on Distributed Objects and Applications, 2001. DOA '01.

[20] Caeiro, M.; Anido, L.; Santos, J.M.; Rodriguez, J. "Towards the standardization of collaborative learning systems" Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2002. WET ICE 2002.

[21] Changtao Qu; Nejdl, W. "Exploring JXTASearch for P2P educational media discovery". Proceedings IEEE Workshop on Knowledge Media Networking, 2002

[22] Qu, C., W. Nejdl, H. Schinzel."Integrating Schema-specific Native XML Repositories into a RDF-based E-Learning P2P Network"Proceedings of the 2nd International Conference on Dublin Core and Metadata Applications (DC 2002).

[23] Wolfgang Nejdl." Semantic Web and Peer-to-Peer-Technologies for Distributed Learning Repositories" 17th IFIP World Computer Congress, Intelligent Information Processing / IIP-2002.

[24] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf ."Content-Based Addressing and Routing: A General Model and its Application". Technical Report CU-CS-902-00, Department of Computer Science, University of Colorado.

E.1.5 Artículos sobre sistemas de enseñanza a distancia y tecnologías relacionadas

[25] Ching-Tang Hsieh; Shih, T.K.; Wen-Chih Chang; Wen-Chieh Ko. "Feedback and analysis from assessment metadata in E-learning". Proceedings of the 17 th International Conference on Advanced Information Networking and Applications. AINA 2003.

[26] Shih, T.K. "Distance education technologies: current trends and software systems" Proceedings of the First International Symposium on Cyber Worlds, 2002.

[27]Zhongnan S., Yuanchun Shi, Guangyou X. "A Learning Resource Metadata Management System Based on LOM Specification". The 7th International Working Group on Computer Supported Cooperative Work in Design (CSCWD2002), Rio de Janeiro, Sep. 2002

[28] Changtao Qu; Nejdl, W. "Towards Interoperability and Reusability of Learning Resource: a SCORM-conformant Courseware for Computer Science Education". Proceedings of the 2nd IEEE International Conference on Advanced Learning Technologies (IEEE ICALT 2002).

[29] Miltiadis D. Lytras, Georgios I. Doukidis 2001. "Value dimension of the e-learning concept. Components and metrics" 20th World Conference on Open Learning and Distance Education (D2001).

[30] Mikael Nilsson, Matthias Palmér, Ambjörn Naeve "SemanticWeb Meta-data for e-Learning – Some Architectural Guidelines".Proceedings of the 11th World Wide Web Conference (WWW2002).